INSIDE MACINTOSH

QuickDraw GX Printing Extensions and Drivers

Addison-Wesley Publishing Company

Reading, Massachusetts Menlo Park, California New York
Don Mills, Ontario Wokingham, England Amsterdam Bonn
Sydney Singapore Tokyo Madrid San Juan
Paris Seoul Milan Mexico City Taipei

ISBN 0-201-40678-0
1 2 3 4 5 6 7 8 9-CRW-9897969594
First Printing, April 1994

Library of Congress Cataloging-in-Publication Data

Inside Macintosh. QuickDraw GX : printing extensions and drivers / Apple Computer, Inc.
p. cm.
Includes index.
ISBN 0-201-40678-0
1. Macintosh (Computer) 2. Computer graphics. 3. QuickDraw GX.
I. Apple Computer, Inc.
QA76.8.M3I5623 1994
005.7' 1265—dc20 93-49875
CIP

Contents

Figures, Tables, and Listings

About This Book

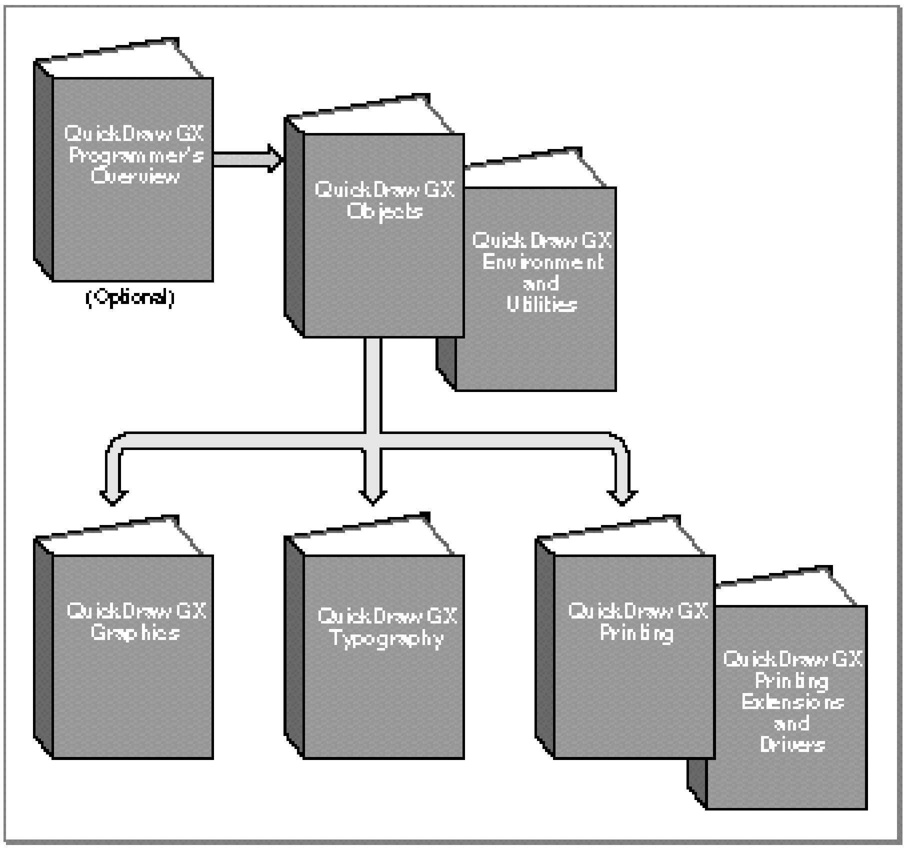
QuickDraw GX is an integrated, object-based approach to graphics programming on Macintosh computers. This book, *Inside Macintosh: QuickDraw GX Printing Extensions and Drivers,* describes how to design and develop printing extensions and printer drivers for use with QuickDraw GX.

For application programming purposes, QuickDraw GX augments the capabilities of some of the Macintosh system software managers documented in other parts of *Inside Macintosh.* Information in this book specifies how to make your printer driver compatible with applications that use the Macintosh Printing Manager, which is described in *Inside Macintosh: Imaging With QuickDraw.*

This book is necessary if you are developing a printing extension or printer driver for use with QuickDraw GX. Before reading this book, you should already be familiar with how QuickDraw GX printing works, as described in *Inside Macintosh: QuickDraw GX Printing.*

For more information about programming with QuickDraw GX, you need to refer to the other QuickDraw GX books in the *Inside Macintosh* suite, including *Inside Macintosh: QuickDraw GX Objects*, *Inside Macintosh: QuickDraw GX Graphics*, and *Inside Macintosh: QuickDraw GX Typography.* Figure P-1 shows the suggested reading order for the QuickDraw GX books. A pictorial overview of *Inside Macintosh,* including the QuickDraw GX suite of books, appears on the inside back cover.

Figure P-1 Roadmap to the QuickDraw GX suite of books


QuickDraw GX Programmer's Overview
(Optional)
QuickDraw GX Objects
QuickDraw GX Environment and Utilities
QuickDraw GX Graphics
QuickDraw GX Typography
QuickDraw GX Printing
QuickDraw GX Printing Extensions and Drivers

What to Read

This book is intended for developers who are interested in enhancing the printing capability of Macintosh systems by writing printing extensions and for developers who are writing printer drivers for use with QuickDraw GX.

Chapter 1, “Introduction to Printing Extensions and Printer Drivers,” provides an overview of the QuickDraw GX printing architecture and how printing extensions and printer drivers work within the QuickDraw GX framework, including a description of how you use printing messages to implement your extensions and drivers.

Chapter 2, “Printing Extensions,” describes what printing extensions are and how they work. This chapter provides a tutorial walk-through of a simple printing extension, including descriptions of the code and resources used to implement the extension.

Chapter 3, "Printer Drivers," describes what printer drivers are and how they work. This chapter provides a tutorial walk-through of a printer driver, including descriptions of the code and resources used to implement the printer.

Chapter 4, "Printing Messages," provides a complete reference to the messages that you can override to implement printing extensions and printer drivers. This chapter also provides a complete reference to the constants and data types that you use with the printing messages.

Chapter 5, "Printing Functions for Message Overrides," provides a complete reference to the QuickDraw GX functions that you can call from within your printing message overrides. This chapter also provides a complete reference to the constants and data types that you use with the printing functions.

Chapter 6, "Printing Resources," provides a complete reference to the resources that you can use to implement printing extensions and printer drivers. This chapter also provides a complete reference to the constants and data types that QuickDraw GX provides to represent the resources.

Chapter Organization

This book contains an introductory chapter, two tutorial chapters, and three reference chapters. Each of the tutorial chapters follows the same structure. For example, the chapter "Printing Extensions" contains these major sections:

- "About Printing Extensions." This section provides an overview of printing extensions.
- "Writing Printing Extensions." This section describes how to develop a printing extension. It uses a detailed walk-through of a sample extension to provide code examples.
- "Using Resources in Printing Extensions." This section describes the resources that you use to implement a printing extension and provides examples of these resources from the sample program.

The three reference chapters follow a standard general structure. For example, the chapter "Printing Functions for Message Overrides" contains these major sections:

- "About The Printing Functions." This sections provides an overview of the printing functions that you can call from within your implementation of a printing message override.
- "Using The Printing Functions." This section describes how you can use the printing messages in your message overrides for various purposes. It describes how to use the most common functions, gives related user-interface information, provides code samples, and supplies additional information.

n “Printing Functions Reference.” This section provides a complete reference to the printing functions that you can use in message overrides by describing the functions along with the constants and data types that you use with them. Each function description follows a standard format, which gives the function declaration; a description of every parameter; the function result, if any; and a list of result codes. Most function descriptions give additional information about using the function and include cross-references to related information elsewhere. Each function description also includes a list of the error codes that can be returned by the function as a result.

n “Summary of Printing Functions.” This shows the C interface for the printing functions and their associated constants and data types.

Conventions Used in This Book

This book uses various conventions to present certain types of information.

Special Fonts

All code listings, reserved words, and the names of data structures, constants, fields, parameters, and functions are shown in Courier (`this is Courier`).

When new terms are introduced, they are in **boldface.** These terms are also defined in the glossary.

Types of Notes

There are several types of notes used in this book.

Note

A note formatted like this contains information that is interesting but possibly not essential to an understanding of the main text. The wording of the title may say something more descriptive than just “Note,” for example, “Terminology Note.” (An example appears on page 2-3.) u

IMPORTANT

A note like this contains information that is especially important. (An example appears on page 2-11.) s

s **W A R N I N G**

Warnings like this indicate potentially serious problems that you should be aware of as you design your application. Failure to heed these warnings could result in system crashes and loss of data. (An example appears on page 3-23.) s

Numerical Formats

Hexadecimal numbers are shown in this format: 0x0008.

The numerical values of constants are shown in decimal, unless the constants are flag or mask elements that can be summed, in which case they are shown in hexadecimal.

Type Definitions for Enumerations

Enumeration declarations in this book are commonly followed by a type definition that is not strictly part of the enumeration. You can use the type to specify one of the enumerated values for a parameter or field. The type name is usually the singular of the enumeration name, as in the following example:

```
enum gxDashAttributes {
   gxBendDash        = 0x0001,
   gxBreakDash       = 0x0002,
   gxClipDash        = 0x0004,
   gxLevelDash       = 0x0008,
   gxAutoAdvanceDash = 0x0010
};
typedef long gxDashAttribute;
```

Illustrations

This book uses several conventions in its illustrations.

In illustrations that show object properties, properties that are object references are in italics.

In order to focus attention on the key part of some drawings, other parts are printed in gray, rather than black.

Objects in diagrams, whether shown with their properties or without, are represented by distinctive icons, such as these:

Shape object
Style object
Ink object

See, for example, Figure 3-2 on page 3-21.

Development Environment

The QuickDraw GX functions described in this book are available using C interfaces. How you access these functions depends on the development environment you are using.

Code listings in this book are shown in ANSI C. They suggest methods of using various functions and illustrate techniques for accomplishing particular tasks. Although most code listings have been compiled and tested, Apple Computer, Inc., does not intend for you to use these code samples in your applications.

This book describes two sample programs in detail. The sample printing extension, the backwash program, draws a background picture on each printing page. The sample printer driver implements QuickDraw GX printing for the ImageWriter II printer. The source code for these programs is found in the Samples folder that is included in the QuickDraw GX release.

Developer Products and Support

APDA is Apple's worldwide source for over three hundred development tools, technical resources, training products, and information for anyone interested in developing applications on Apple platforms. Customers receive the quarterly *APDA Tools Catalog* featuring all current versions of Apple development tools and the most popular third-party development tools. Ordering is easy; there are no membership fees, and application forms are not required for most of our products. APDA offers convenient payment and shipping options, including site licensing.

To order products or to request a complimentary copy of the *APDA Tools Catalog*, contact

APDA
Apple Computer, Inc.
P.O. Box 319
Buffalo, NY 14207-0319

Telephone	800-282-2732 (United States) 800-637-0029 (Canada) 716-871-6555 (International)
Fax	716-871-6511

AppleLink	APDA
America Online	APDAorder
CompuServe	76666,2405
Internet	APDA@applelink.apple.com

If you provide commercial products and services, call 408-974-4897 for information on the developer support programs available from Apple.

C H A P T E R 1

Introduction to Printing Extensions and Drivers

Contents

This chapter introduces the basic features of printing extensions and printer drivers and provides you with an overview of how printing works with QuickDraw GX. This chapter describes the printing process from the perspective of a printing-extension or printer-driver developer. To understand how printing works from the perspective of an application developer, you need to read *Inside Macintosh: QuickDraw GX Printing.*

Before reading this chapter, you should be familiar with the QuickDraw GX environment and QuickDraw GX objects. These features are described in *Inside Macintosh: QuickDraw GX Environment and Utilities* and *Inside Macintosh: QuickDraw GX Objects.*

This chapter begins with an overview of QuickDraw GX printing extensions and printer drivers. Next, this chapter describes how QuickDraw GX

- uses a message-passing architecture to accomplish printing
- supports three print imaging systems
- performs printing in several phases
- allows you to add panels to print dialog boxes
- uses collections for data storage and access
- provides messages that you can override to create drivers and extensions
- allows you to use resources to define a large portion of the functionality of extensions and drivers

Note

This chapter and book use the term **printer driver** to refer to the driver of any printing device, such as a printer, plotter, or imagesetter. u

About QuickDraw GX Printing Extensions and Printer Drivers

Printing extensions are add-on software components that you can create to extend the printing capabilities of applications. Printing extensions are used for tasks such as supporting hardware additions and modifying the appearance of printed pages and allow you to provide these capabilities without having to write an entire printer driver. When a user places a printing extension in the Extensions folder (which is in the System Folder), the extension is available for use.

Printer drivers translate the instructions that compose QuickDraw GX shapes and pictures into printed output on a specific output device. Each printer driver sends data and instructions in a form specific to the device that it drives and manages the physical communications with that device. You need to develop a separate driver for each hardware device that has different characteristics. Whenever you create a desktop printer, the Macintosh system software loads and uses the driver for that printer.

QuickDraw GX printing extensions and printer drivers are largely data-driven and take advantage of the QuickDraw GX message-based printing architecture, which is briefly described in the next section, “QuickDraw GX Printing and Messages.”

QuickDraw GX Printing and Messages

Printing with QuickDraw GX is based on a **message-passing architecture.** During the printing process, certain printing-related tasks often need to be done, or certain printing-related conditions arise. QuickDraw GX sends a large number of messages to notify such programs as an application, printing extension, and printer driver about these tasks and conditions. These messages are called **printing messages.**

Your printing extension or printer driver might need to respond to these printing messages to make available its features or functionality. For example, your printing extension could respond to a particular printing message called `GXDespoolPage` to add a date-time stamp to each printed page. Or your printer driver could respond to the `GXOpenConnection` message to verify that the corresponding printing device is working properly. This response of intercepting a specific printing message and taking some action is called a **message override,** which is performed by an override function that you define in your extension or driver.

Before you learn more about how message overrides work, it’s important to have an overview of what happens when QuickDraw GX sends a printing message. Because your printing extension or printer driver may respond to a printing message, it is called a **message handler.** A number of message handlers, including your extension or driver, an application, and QuickDraw GX itself, can respond to printing messages.

Each message handler is part of the **message chain,** which links the handlers in a hierarchical sequence. Each handler in the hierarchy receives the message and decides whether to respond to the message and whether to forward the message to the handler below it in the hierarchy. A message handler provides a function to respond to any message that the handler wants to act upon.

QuickDraw GX provides a function to handle each message that it sends. These functions are called the **default implementations** of the messages and are activated whenever a message is not completely handled by another message. When you provide a function to handle a message in your printing extension or printer driver, you implement an **override function,** which can add to or replace the actions provided by other handlers of the message.

The declaration of each override function must exactly match the declaration of the message that it is overriding. You can use any name that you want for the override function, but you must declare the same return value and parameter types as are used for the message declaration. For example, one of the printing messages is named `GXCountPages` and has the following declaration:

```
OSErr GXCountPages(gxSpoolFile aSpoolFile, long *numPages);
```

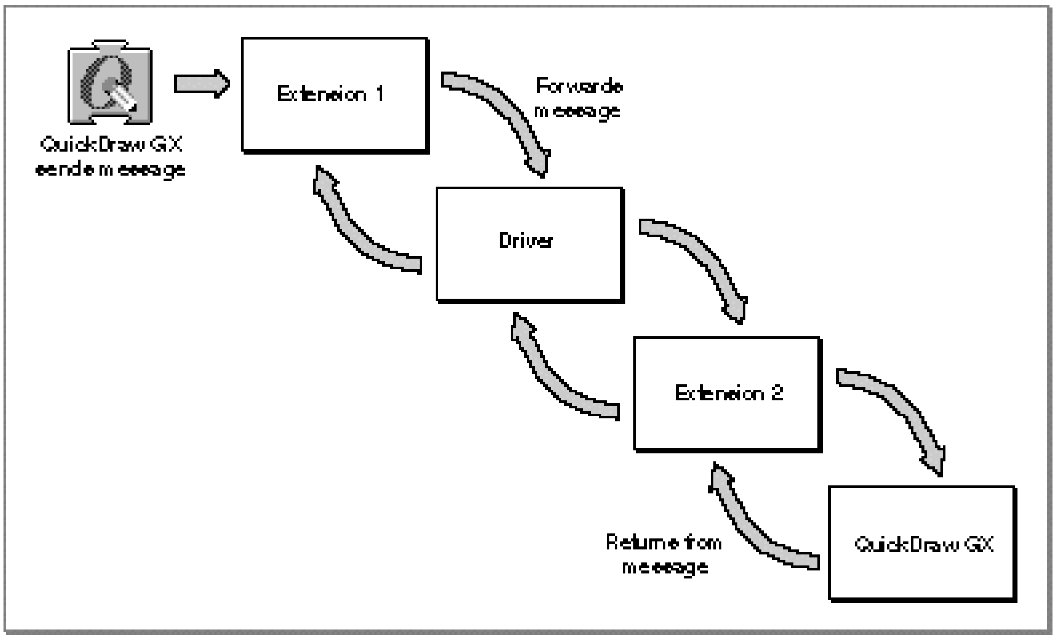
If you override this message, you need to declare your override function with a matching header. For example,

```
OSErr MyCountPages(gxSpoolFile myFile, long *numPages);
```

In most cases, when your printing extension or printer driver overrides a printing message, it performs some operations and forwards the message to the next handler in the message chain. This is called a **partial override.** (A **total override** is when the message is not forwarded.) Although some of the default implementations of the printing messages provided by QuickDraw GX are empty and simply return, most of the default implementations do provide significant functionality. In many cases, you must allow the default implementation of a message to provide its actions; otherwise, a vital operation might be neglected, potentially resulting in serious errors.

A typical message chain, in which several handlers respond to and forward a message, is shown in Figure 1-1.

Figure 1-1 A printing message chain



Each printing message is described in the chapter “Printing Messages” in this book. Each description includes information about whether you must forward the message to allow the default implementation to provide its actions. Each description also specifies whether, for a partial override of a message, you need to forward the message before adding your own actions or after adding your own actions.

The timing of when you forward a message, relative to when you perform your actions in an override function, can result in significant differences. For example, if you are creating a printing extension that combines eight pages of a document into one page of thumbnail sketches, you might override the `GXCountPages` message, which QuickDraw GX sends to count the pages in the spool file. You would forward this message to allow the default implementation to count the pages, and your override would then modify that value. On the other hand, if you are creating an extension that adds a confidential stamp to each page as it gets spooled, you might override the

`GXSpoolPage` message. You would add your stamp to the page before forwarding the message to the default implementation, which would spool the page to file.

QuickDraw GX also provides a number of printing functions that you can only call from within the message overrides that you implement in your extension or driver. These functions perform a variety of operations, including

- displaying status information and the printing alert boxes to the user
- interfacing with the paper trays
- communicating imaging options between a driver and an application
- gracefully handling errors that arise during the processing of certain printing messages

The printing functions that you can call from within your message overrides are described in the chapter “Printing Functions for Message Overrides” in this book.

The message-passing architecture used for QuickDraw GX printing is supported by the Message Manager, which is a general-purpose software component of QuickDraw GX that you can use for message-passing in your own programs. The Message Manager and the concepts and terminology of message-based programming are described in *Inside Macintosh: QuickDraw GX Environment and Utilities.*

Print Imaging Systems

Some of the printing messages that QuickDraw GX sends are specific to the print imaging system that a printing device uses. A **print imaging system** is a part of the QuickDraw GX printing software that manages the conversion of QuickDraw GX shapes into data for a specific type of output device, such as a printer. QuickDraw GX supports four print imaging systems for printing documents, as shown in Table 1-1.

Table 1-1 Print imaging systems that QuickDraw GX supports

Imaging system	Explanation
Raster	For use with a raster output device such as an ImageWriter printer. Raster bitmap data and escape sequences are sent to accomplish the printing of each page.
PostScript™	For use with a PostScript output device such as an Apple LaserWriter printer. PostScript printing instructions are sent to accomplish the printing of each page.

Table 1-1 Print imaging systems that QuickDraw GX supports (continued)

Imaging system	Explanation
Vector	For use with a vector output device that uses a plotting language such as HPGL. QuickDraw GX shapes are converted into vectors, and the vector data and pen information are sent to the device to accomplish the printing of each page.
Portable digital document	For the creation of a portable digital document (PDD).

NOTE If you are writing a driver that creates portable digital documents for transportation to other types of computing systems, you specify that your driver works with the portable digital document (PDD) imaging system, which is described in *Inside Macintosh: QuickDraw GX Environment and Utilities.*

QuickDraw GX defines a few imaging messages for the raster and vector imaging systems and a large number of imaging messages for the PostScript imaging system. For each message, QuickDraw GX provides a default implementation that performs the basic task for which the message is intended. You can override any of these messages to customize the behavior for your extension or driver. In the resources that you provide, you specify which imaging system your extension or driver uses and which messages you are overriding.

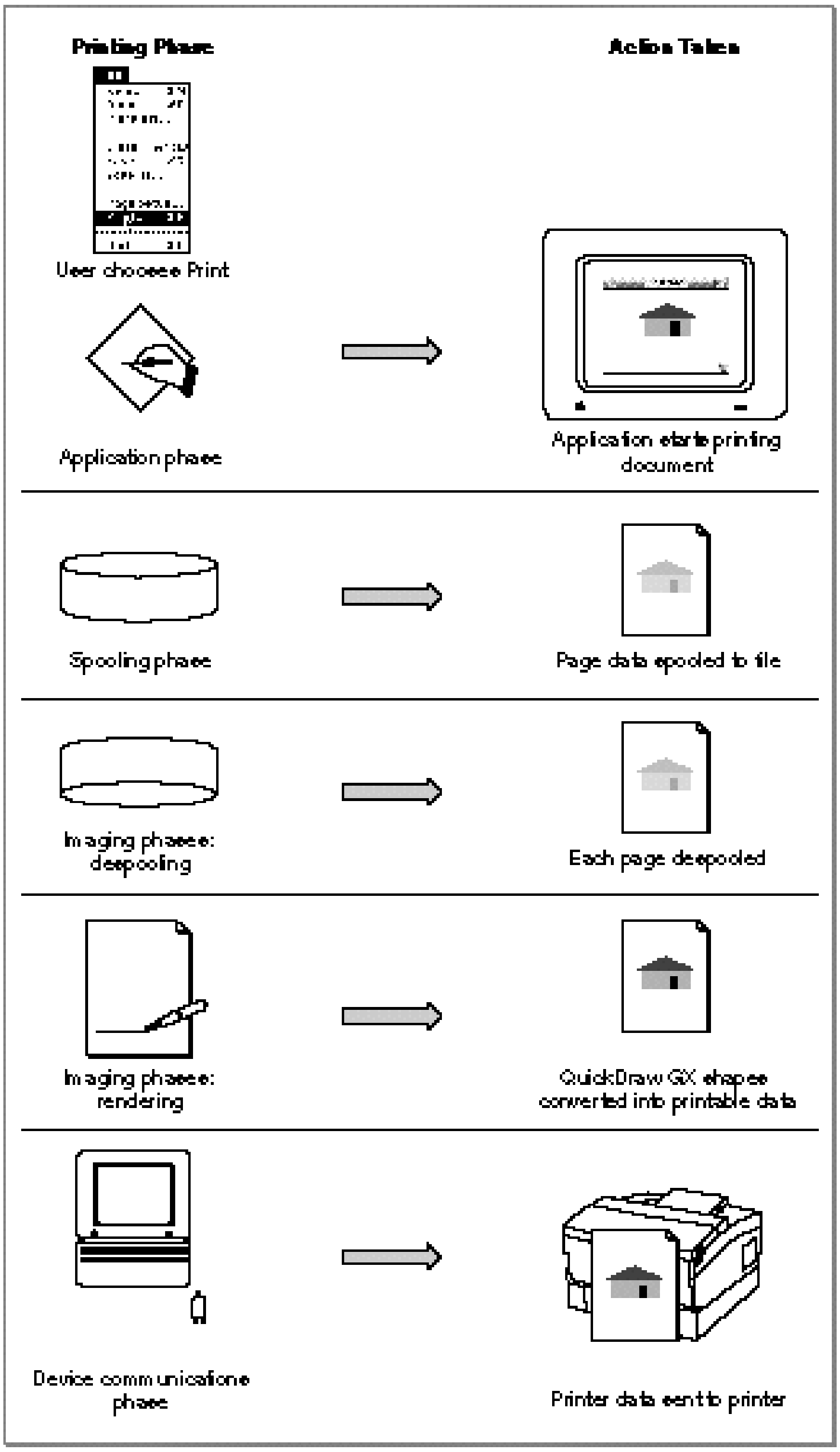
When you develop a printing extension, you can choose to make the extension available for any or all of the imaging systems, depending on what tasks the extension needs to perform. When you develop a printer driver, you specify the appropriate imaging system for that printer. For example, if you are writing a raster printer driver, you specify that your driver works with the raster imaging system. You can then use the raster package controls (`'ropt'`) resource to define the escape sequences used for performing line feeds on the printer. You can also override the `GXRasterLineFeed` message to implement how a line feed is performed.

If you are writing a PostScript printer driver, you specify that your driver works with the PostScript imaging system. You can then use the PostScript preferences (`'pdip'`) resource to specify such things as which level of PostScript and which color space your driver supports. You can also override any of the numerous PostScript printing messages to exercise control over your device.

Printing Phases

QuickDraw GX sends specific printing messages during each of the four phases of printing. Figure 1-2 shows what happens to document data in these phases of printing.

Figure 1-2 The phases of QuickDraw GX printing


Printing Phase
Action Taken
User chooses Print
Application phase
Application starts printing document
Spooling phase
Page data spooled to file
Imaging phase: despooling
Each page despooled
Imaging phase: rendering
QuickDraw GX shapes converted into printable data
Device communications phase
Printer data sent to printer

The four phases of printing are:

- The **application phase** of printing, during which the application calls QuickDraw GX and interacts with the user by displaying dialog boxes to establish printing parameters, such as page orientation and paper type. Examples of messages sent during this phase are `GXStartJob`, `GXPrintPage`, and `GXFinishJob`.
- The **spooling phase** of printing, during which the application spools the document pages to disk, in preparation for printing. QuickDraw GX sends messages during this phase to notify you when each page is about to be spooled. Examples of messages sent during this phase are `GXCreateSpoolFile`, `GXSpoolPage`, and `GXSpoolData`.
- The **imaging phase** of printing, during which each previously spooled page is rendered into a form that can be printed on the output device.

  The imaging phase is actually composed of two processes:

  - **Despooling,** during which each previously spooled page is read from the spool file. Examples of the despooling messages are `GXCountPages`, `GXDespoolPage`, `GXDespoolData`, `GXDespoolResource`, and `GXCloseSpoolFile`.
  - **Rendering,** during which each despooled page is converted into image data that can be printed by the output device. Some rendering messages, known as universal imaging messages, are sent for all imaging systems. Examples of these are the `GXImageJob`, `GXImagePage`, and `GXRenderPage` messages. Other messages sent during rendering depend on which imaging system is in use.

    Examples of raster rendering messages are `GXRasterDataIn`, `GXRasterLineFeed`, and `GXRasterPackageBitmap`. Examples of PostScript rendering messages are `GXPostScriptQueryPrinter`, `GXPostScriptInitializePrinter`, `GXPostScriptStreamFont`, and `GXPostScriptDoPageSetup`. Examples of vector rendering messages are `GXVectorPackageShape`, `GXVectorLoadPens`, and `GXVectorVectorizeShape`.
- During the **device communications phase** of printing, the data that represents the rendered form of each page is sent to the output device. Messages sent during this phase include `GXOpenConnection`, `GXStartSendPage`, `GXWriteData`, and `GXCheckStatus`. You can only communicate with the printing device during this phase of printing, which means that you must override the device communications messages to communicate with the user regarding printer supplies such as paper.

The four phases of printing can be, but are not necessarily, sequential. In some cases, these phases are interleaved. For example, when the same computer is performing both formatting and output of a document, the application and spooling phases are interleaved, and the imaging and device communications phases are interleaved. The application initiates the printing of each page, and that output is written to a spool file. When the printer is ready, each page is despooled, rendered, and sent to the device.

Printing extensions and printer drivers can override any of the messages that QuickDraw GX sends during each of these printing phases, allowing you a tremendous degree of flexibility in controlling printing on a specific device.

Note
The spooling, imaging, and device communications phases of printing can occur on different devices. For example, an application can spool a document to a printer server on a network, which might then image the document to a disk that is taken to a printing service and printed on a high-resolution printer. u

Extensions, Drivers, and the User Interface

QuickDraw GX sends printing messages when the user chooses commands that display such print dialog boxes as the Page Setup, Custom Page Setup, and Print dialog boxes. Your printing extension or printer driver can override the appropriate printing message to add one or more panels to one of these print dialog boxes. These panels become available to the user when the user clicks the More Choices button in the dialog box. When the user opens a panel that you added, your extension or driver can continue to receive and respond to messages as the user manipulates the panels, including messages that notify you if a user closes a panel or if a user confirms or cancels the print dialog box.

Much of the information that a user specifies in a print dialog box is stored in one of the collections that QuickDraw GX provides: the job collection, the format collection, or the paper-type collection. These collections add to the printing information already available in the corresponding print objects: the job object, the format object, and the paper-type object. Collections also provide data extensibility for printing extensions and printer drivers. For example, you can use the Collection Manager to create your own collection items for storing data related to the options a user chooses in a panel that your extension or driver adds to a dialog box. An example of how an extension uses a collection is in the chapter “Printing Extensions” in this book.

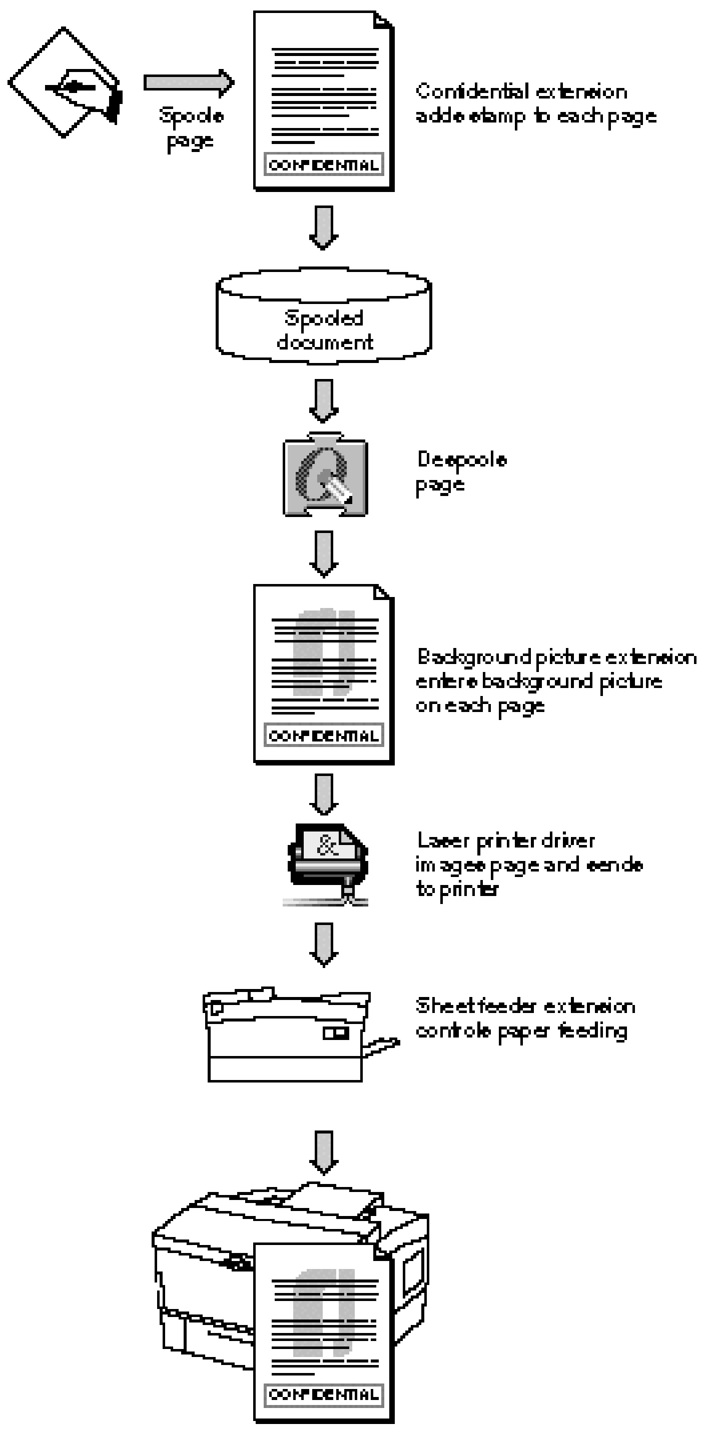
Adding panels to a dialog box is described in *Inside Macintosh: QuickDraw GX Printing*, and the messages that QuickDraw GX sends when a print dialog box is displayed or an event happens in a panel are in the section “Dialog Box Messages” beginning on page 4-81 in the chapter “Printing Messages.” Information about manipulating the job, format, and paper-type collections is in *Inside Macintosh: QuickDraw GX Printing*. Information about using the Collection Manager is in *Inside Macintosh: QuickDraw GX Environment and Utilities.*

Using Printing Extensions and Printer Drivers Together

While printer drivers are device dependent by definition, printing extensions can be device independent, which allows you to develop printing features that can be used with any number of devices.

The printing of a document always involves exactly one or two printer drivers: the formatting driver and the output driver, as described in *Inside Macintosh: QuickDraw GX Printing*. In many cases, formatting and output are handled by the same printer driver. However, any number of different printing extensions can be active at the same time. For example, you could create one extension that controls a sheet feeder that can be connected to numerous printers, another extension that draws a background picture on each printed page, and another extension that prints a confidential stamp on each page. These extensions could all be active at the same time as one printer driver, as shown in Figure 1-3.

Figure 1-3 Using multiple extensions concurrently


Spools page
CONFIDENTIAL
Confidential extension adds stamp to each page
Spooled document
Despools page
CONFIDENTIAL
Background picture extension enters background picture on each page
Laser printer driver images page and sends to printer
Sheet feeder extension controls paper feeding
CONFIDENTIAL

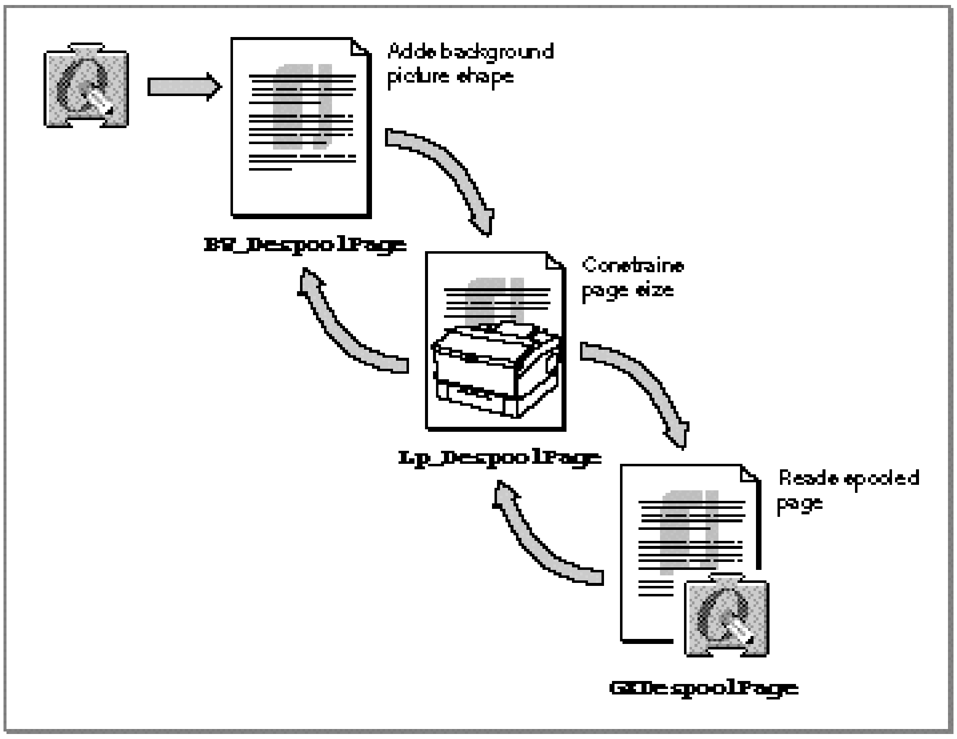
In Figure 1-3, the confidential stamp extension adds a confidential stamp at the bottom of the page. This extension overrides the `GXSpoolPage` message and adds the shape object that represents the confidential stamp to the shapes that compose the page when the page is sent to the spool file.

The background picture extension mentioned in Figure 1-3 adds a panel to the Print dialog box to allow the user to select a picture file to use. The user can choose not to have a background picture or can select a file. If the user selects a picture file, this extension overrides the `GXDespoolPage` message so that it can draw the background picture after the page has been despooled and before it has been sent to the printer. The background picture extension is described extensively in the chapter "Printing Extensions" in this book.

The sheet feeder extension mentioned in Figure 1-3 works with a sheet-feeding device that can be added to laser printers produced by a number of manufacturers. This extension adds a panel to the Print dialog box to allow the user control over the sheet feeder and overrides several paper-handling messages to interface with the sheet feeder.

The laser printer driver that is mentioned in Figure 1-3 overrides a number of printing messages, including the `GXDespoolPage` message, which the background printing extension also overrides. The flow of control for the `GXDespoolPage` message in this case is shown in Figure 1-4, which includes only those message handlers that override the message.

Figure 1-4 Several functions handling the `GXDespoolPage` message



Whenever QuickDraw GX sends the `GXDespoolPage` message, as in this example, the following flow of control results.

1. The confidential stamp extension doesn't override this message, so QuickDraw GX doesn't send it to this extension.
2. The background picture extension overrides `GXDespoolPage` with the `BW_DespoolPage` function, which forwards the message before modifying the page.
3. The sheet feeder extension doesn't override this message, so QuickDraw GX doesn't forward it to this extension.
4. The laser printer driver overrides `GXDespoolPage` with the `Lp_DespoolPage` function, which forwards the message before modifying the page.
5. The QuickDraw GX default implementation of `GXDespoolPage` reads the graphics and formatting data for the page.
6. The laser printer driver override regains control and performs its operations to restrict the page size to the size supported by the device.
7. The background picture extension override regains control and performs its operations to add the background picture to the picture shape that represents the page.

One of the important issues that you have to decide when overriding a message in a printing extension or printer driver is whether to forward the message before adding your own code or after adding your own code. The timing of when your operations are performed can be critical, as is the case in the previous example. The chapter "Printing Messages" contains information about each printing message, which can help you to decide when timing might be an issue.

Any applications that use QuickDraw GX printing features can take advantage of extensions without any coding: the user controls whether extensions are applied to the printing of documents. In the example shown in Figure 1-3, the printed output of any application that prints while the extensions are enabled will use the background picture, confidential stamp, and selected paper-feeding options.

Using Resources to Create Printing Extensions and Printer Drivers

Printing extensions and printer drivers are very similar entities. Each contains a number of resources and overrides a certain number of the printing messages to provide its functionality. Each printing extension and printing driver includes resources for such diverse purposes as

- displaying extension, driver, and desktop printer icons
- displaying dialog box panels and defining controls
- specifying of which kind of device communications your driver uses, such as serial or PAP
- specifying which messages the extension or driver overrides

- defining when an extension needs to be loaded into memory
- specifying the imaging system (raster, vector, PostScript, or portable digital document) that your printer driver uses
- defining imaging options such as color and grayscale control
- defining status and aler,t conditions and displaying information about them

All of the resources that you need to define for printing extensions and printer drivers are described in the chapter “Printing Resources” in this book. Some resources are required for both extensions and drivers, some are required for only one, and some are optional.

QuickDraw GX allows you to define much of the functionality of your extensions and drivers in these printing resources. Given that QuickDraw GX also provides default implementations for most of the printing messages, you can develop many printing extensions and printer drivers without writing much code. In fact, some very useful printing extensions (including the confidential stamp extension) and some PostScript printer drivers have been written by overriding only one or two printing messages.

Overriding Printing Messages

When you decide that you need to override a certain printing message in a printing extension or printer driver, you need to follow these three steps:

1. Write the code to implement your override. You can name your override function with whatever name you want. You need to decide whether your override is going to forward the message to other message handlers (partially override) or not forward the message at all (totally override). If you are partially overriding the message, then you must decide at which point in your code to forward the message to other handlers.
2. Add the message name to one of the override (`'over'`) resources that you include with your extension or driver. Each entry in an override resource specifies the QuickDraw GX message ID, the code segment, and the offset of the jump statement to that function in the jump table. The override resource is described in the chapter “Printing Resources” in this book.
3. Add a jump statement for your message override to the assembly-language jump table that you must include in the code for your extension or driver.

The chapters “Printing Extensions” and “Printer Drivers” in this book provide examples of message overrides and override resources and describe the jump table.

Defining Components of the User Interface

You need to manage some user interface features in most printing extensions and all printer drivers. You can define most of these features in resources; however, there are some tasks that you need to implement in code.

When creating a printing extension, you need to define icons for your printing extension, you might need to manage the display of status information or printing alert boxes, and you might need to add panels to one or more of the print dialog boxes.

When creating a printer driver, you need to define icons for your printer driver, you must provide a Chooser interface, and you must manage the reporting of status and alert conditions to the user while printing is in progress. You display status information to which the user need not respond, and you display printing alert boxes to the user when a condition arises that requires a response, such as a paper jam. All of these tasks are described in the chapter “Printer Drivers” in this book.

In addition, you may want to add a panel to a print dialog box to provide options and controls to the user. Most printer drivers add panels to allow the user to specify options such as quality control or paper-type assignments. You can read about adding panels in your extensions and drivers in the chapters “Printing Extensions” and “Printer Drivers” in this book. You can also read about adding panels in *Inside Macintosh: QuickDraw GX Printing*.

Planning How to Write a Printing Extension or Printer Driver

You use similar development strategies for writing both printing extensions and printer drivers, including the following steps:

- Determine which imaging system(s) you are using.
- Determine which printing messages you need to override.
- Write the code to override those messages.
- Define the resources that provide data for QuickDraw GX to use with your extension or driver.

The chapter “Printing Extensions” in this book provides a detailed description of developing a printing extension. The chapter “Printer Drivers” in this book provides a detailed description of developing a printer driver.

The chapter “Printing Messages” in this book provides a complete reference guide to all of the printing messages that you can override to implement your printing extensions and printer drivers. The chapter “Printing Functions for Message Overrides” in this book provides a reference guide to all of the printing functions that you can call from within your message overrides. The chapter “Printing Resources” in this book provides a reference guide to all of the resources that you use to write your printing extensions and printer drivers.

C H A P T E R 2

Printing Extensions

Contents

This chapter describes how you can develop printing extensions to modify or add to the printing features provided by QuickDraw GX. You need to read this chapter if you are developing printing extensions, including those that support a hardware device or modify the appearance of the pages that are printed by applications.

To use this chapter, you need to be familiar with how the printing features in QuickDraw GX work. *Inside Macintosh: QuickDraw GX Printing* describes these features and gives you an overview of the printing process. The chapter "Introduction to Printing Extensions and Drivers" in this book provides an overview of how QuickDraw GX provides printing and how you can override portions of its functionality in a printing extension or printer driver.

Printing extensions comprise a collection of resources. Before reading this chapter, you need some understanding of Macintosh resources and resource files. You can read about the Macintosh Resource Manager in *Inside Macintosh: More Macintosh Toolbox*.

Printing extensions make use of collections, which are managed by the Collection Manager, and are activated by messages, which are managed by the Message Manager. You need to know about the Collection Manager and Message Manager components of QuickDraw GX to understand how extensions work. Both of these managers are described in *Inside Macintosh: QuickDraw GX Environment and Utilities.*

This chapter begins with an overview of printing extensions and then describes

- the kinds of tasks that you can use a printing extension to perform
- the development of a printing extension
- the resources that compose a printing extension
- some of the messages that you can override to develop a printing extension
- the interactions between printing extensions and QuickDraw GX

This chapter gives you the conceptual information that you need in the form of a guided tour through an extension that draws a background picture on each page of a document. For specific reference information about the messages that you override to write a printing extension, see the chapter "Printing Messages" in this book. For specific reference information about the resources that you need to include in your printing extension file, see the chapter "Printing Resources" in this book.

Note
The code samples presented in this chapter are taken from a recent version of the sample code. These samples are not complete and may have been slightly modified since printing. You can find the latest version of the code in the QuickDraw GX sample code. u

About Printing Extensions

Printing extensions are add-on software components that you can develop to extend the printing capabilities of QuickDraw GX. Printing extensions are used for tasks such as supporting hardware additions and modifying the appearance of printed pages and allow you to provide these capabilities without having to implement an entire driver. For example, you could develop an extension that allows users to control a sheet feeder on a printer or that adds a confidential stamp to every page produced by an application.

Just as Macintosh system software extends its functionality with system extensions and control panels, QuickDraw GX allows you to extend the functionality of printing by providing printing extensions. You can

- add panels to the QuickDraw GX print dialog boxes
- obtain additional information from users to control options within your extension
- modify the manner in which each page, or a whole document, is printed
- drive hardware extensions to printing devices

This added functionality is completely independent of the mainstream execution of an applications, and users can activate an extension at the time they run an application. Active printing extensions reside in the Extensions folder inside the System Folder, along with system extensions and control panels. Unlike system extensions and control panels, which run at system startup time, printing extensions run at print time. Any number of printing extensions can be active for a given print job.

Application developers can create applications that initiate printing functions without knowing whether printing extensions will run with the application. When a printing extension is active, QuickDraw GX adds the activity of the extension to that of the requested printing function. If no extensions are active, QuickDraw GX just performs the requested printing function. QuickDraw GX takes care of locating and activating the printing extensions at the appropriate time. All you have to do when you develop the extension is to provide QuickDraw GX with information about how and when you want the extension used.

Since printing extensions run independently of applications, a single extension can run with a number of different applications. Thus, for example, you can write an extension that adds a special stamp to a printed page, and users can activate it whenever they want that stamp added to their printed documents.

Printing extensions also allow you to write a single piece of code to handle hardware add-on devices rather than rewriting a driver to do it. By doing this, you provide users with the ability to support the hardware add-on for multiple printers. Your extension can add a panel to one of the print dialog boxes to extend control of the device to the user. For example, the same extension can drive a sheet feeder device that you can attach to a number of different printers.

Printing Extension Tasks

Printing extensions can affect printing at well-defined points during the printing process. QuickDraw GX searches for active extensions and sends messages to them to allow you to extend printing capabilities. You can

- add panels to the Page Setup, Custom Page Setup, and Print dialog boxes and monitor the user's activity in those panels
- modify printed pages during the spooling phase of printing
- modify printed pages during the imaging phase of printing
- modify the manner in which document pages are sent to the printer during the device communications phase of printing

Each of these extension capabilities is described in the sections that follow.

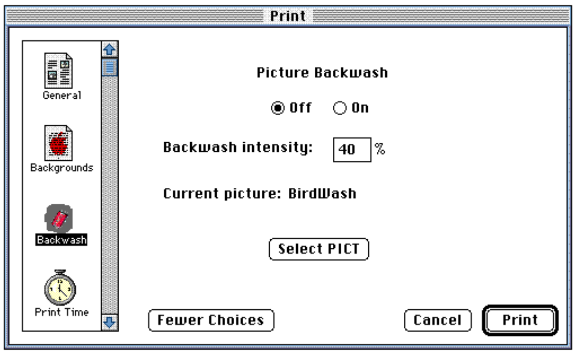
Printing phases are described in the chapter "Introduction to Printing Extensions and Drivers" in this book. The dialog boxes used during the printing process are described in detail in *Inside Macintosh: QuickDraw GX Printing*. The chapter "Printing Messages" in this book describes each of the messages that QuickDraw GX sends during the printing process.

Adding Panels to Print Dialog Boxes

QuickDraw GX allows applications, extensions, and drivers to add panels to the print dialog boxes. Adding a panel and monitoring the user's selections in the dialog box is described in *Inside Macintosh: QuickDraw GX Printing*. While the user manipulates the panel, QuickDraw GX sends event messages (the `GXFilterPanelEvent` and the `GXHandlePanelEvent` messages) to your extension. It also sends messages before closing a panel and notifies you whether the user confirmed or canceled the dialog box. The `GXFilterPanelEvent` message is described on page 4-86 and the `GXHandlePanelEvent` message is described on page 4-85 in the chapter "Printing Messages."

You can add panels for users to control the capabilities that your printing extension provides. For example, the background picture printing extension adds a panel to the expanded Print dialog box that is displayed when the user clicks the More Choices button in the Print dialog box. This panel allows the user to specify which picture file to use as the background picture on printed pages, as shown in Figure 2-1.

Figure 2-1 The background picture panel displayed in the Print dialog box


Print

Picture Backwash

[✓ Off] On

Backwash intensity: 40 %

Current picture: BirdWash

Modifying a Page During Spooling

You can develop a printing extension to make device-independent changes to image data during the spooling phase of printing. QuickDraw GX sends a message to the extension just before each page in a document is spooled to disk. Your printing extension can take this opportunity to make any changes to the page that it wishes. The changes are then incorporated into the final version of the page that is saved in the spool file.

For example, you might develop an encryption printing extension that is used during spooling to encode the printing data as it is spooled to disk and decode it as it is read for printing. This extension allows security-conscious organizations to print sensitive documents and ensure that they are secure during the time they reside in spooler files. It also allows the spooled files to be distributed over a network with confidence that the contents are hidden during transmission time.

Modifying a Page During Imaging

You can develop a printing extension to modify a page in either a device-independent or device-dependent manner during the imaging phase of printing. QuickDraw GX sends messages to the extension just after opening the spool file for imaging and just before closing the spool file at the completion of imaging. Examples of printing extensions that modify the page are:

- A confidential stamp printing extension that stamps the word “Confidential” across each page of a document as it is printed. You can set up a shared confidential printing station with this extension attached to it, and any document printed to this printing station will have the confidential stamp added to its pages.
- A background picture printing extension that gives a user a choice of backgrounds to print against. For example, a user could use this extension to add color ramps to a document being printed as slides for a presentation. Or a user could specify a picture file to use as a background picture, as shown in Figure 2-2 on page 2-8.
- A thumbnail printing extension that reduces the page size so that a number of logical pages can be printed together on one physical page. This is useful for proofing and creating story boards.

Modifying the Device Communications Process

You can develop a printing extension to modify the manner in which each printed page is sent to the printer. For example, you can develop an extension that creates a PostScript file copy of each document that is printed. This extension overrides the printing message that sends data to the printer (the `GXDumpBuffer` message, which is described on page 4-142 in the chapter “Printing Messages”) and copies the buffer to a file.

Developing a Printing Extension

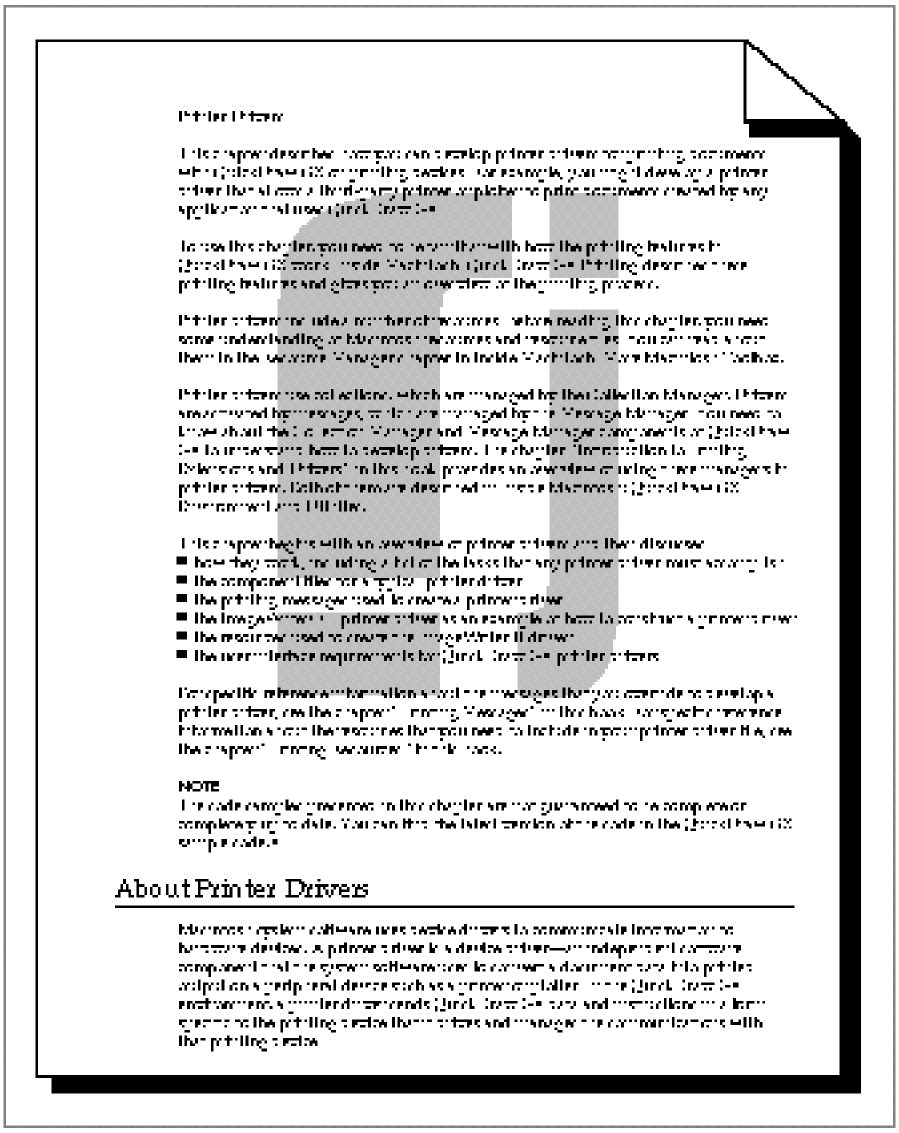
The composition of your printing extension depends on the capabilities that you are adding to QuickDraw GX printing. Each printing extension must at least include a resource file, an assembly-language jump table, one or more files of code to implement the functions that your extension perform. The background picture printing extension is typical; it consists of four files, as shown in Table 2-1. The rest of this chapter describes the contents of each of these files. The complete files are found in the QuickDraw GX sample code.

Table 2-1 Files used to implement the background picture printing extension

Filename	**Contents**
`backwash.a`	The assembly-language jump table that QuickDraw GX uses to locate the message overrides for the background picture printing extension
`backwash.h`	The C language header file that includes all of the other needed header files, defines the constants and data types used by the program, and references external items
`backwash.c`	The C language code that implements the message overrides for the background picture printing extension
`backwash.r`	The resources for the background picture printing extension

The background picture printing extension draws a background picture on each page as the page is despooled, just before it is imaged on the output device. A page printed while the background picture printing extension is active can have any picture printed on it. An example is shown in Figure 2-2.

Figure 2-2 A page printed while the background picture printing extension is active



Printer Drivers

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]

[illegible]

NOTE

[illegible]

About Printer Drivers

[illegible]

The background picture printing extension also adds a panel to the Print dialog box. The panel allows the user to enable or disable the extension, to select which background picture to use, and to set an intensity level for drawing the picture on the page. The Print dialog box with the background picture panel added is shown in Figure 2-1 on page 2-5.

The Jump Table

You need to tell QuickDraw GX where (in which segment and at what offset from the beginning of that segment) to find the code for each printing message that you are overriding. You do this by defining an assembly-language jump table. The contents of the file `backwash.a`, which defines the jump table for the background picture printing extension, are shown in the QuickDraw GX sample code.

The jump table links the appropriate override function into your code and provides a jump statement to invoke that function. The override resource tells QuickDraw GX where to look (at which offset) in the jump table to find the jump statement for a specific printing message name. In this way, QuickDraw GX knows which of your override functions to call to respond to the messages in which you are interested.

You define a jump table with one jump (`JMP`) statement for each message that you override. You also define an override resource that specifies the offset in that jump table for each message. The jump table and override resource must be coordinated. QuickDraw GX dispatches a printing message to your extension by jumping to the routine whose location is specified in the appropriate location in the jump table. The override resource is described in the section “The Override ('over') Resource” beginning on page 6-13 in the chapter “Printing Resources.”

For example, the background picture printing extension overrides several printing messages, as shown in Table 2-2.

Table 2-2 Printing messages overridden by the background picture printing extension

Message	Why you override
`GXInitialize`	To set up the environment so that the extension can use predefined global values
`GXShutDown`	To free the storage that was allocated by the `GXInitialize` message override
`GXJobPrintDialog`	To add a panel to the Print dialog box so the user can select the background picture filename and intensity
`GXHandlePanelEvent`	To handle events that occur in the panel
`GXCreateSpoolFile`	To add the background picture to the spool file
`GXDespoolPage`	To draw the background picture on each page as it is despooled
`GXCloseSpoolFile`	To free the storage allocated for the background picture shape

The override resource for the background picture printing extension has to include an entry for each of these printing messages, and the jump table has to include a jump statement for each. Listing 2-1 shows the override resource definition from the `backwash.r` file. You can read about each of the printing messages in the chapter “Printing Messages” in this book.

Listing 2-1 The override resource for the background picture printing extension

```
resource gxOverrideType (gxExtensionUniversalOverrideID,sysHeap,
                                                     purgeable)
{
   {
   gxInitialize,        0, 4,
   gxShutDown,          0, 8,
   gxJobPrintDialog, '  0, 12,
   gxHandlePanelEvent,  0, 16,
   gxCreateSpoolFile,   0, 20,
   gxDespoolPage,       0, 24,
   gxCloseSpoolFile,    0, 28
   };
};
```

The name of each message is followed by the ID of the extension's code segment in which its code resides and the byte offset of its jump statement in the jump table. QuickDraw GX reserves the first 4 bytes for its own use, which makes 4 the first offset that you can use. These 4 bytes are used by QuickDraw GX to maintain an owner count and must be set to 0 in the jump table.

You implement the jump table as an assembly-language program that contains a jump statement for each override function. You need to list the jump statements with exactly the same offsets as you listed for the message names in your override resource; otherwise, QuickDraw GX will invoke the wrong function in response to a message. Listing 2-2 shows the jump table for the background picture printing extension.

Listing 2-2 The jump table for the background picture printing extension

```
   EXPORT EntryPoint
   IMPORT BWInitialize
   IMPORT BWShutDown
   IMPORT BWJobPrintDialog
   IMPORT BWHandlePanelEvent
   IMPORT BWCreateSpoolFile
   IMPORT BWDespoolPage
   IMPORT BWCloseSpoolFile

EntryPoint  PROC                ; main entry point

   DC.L     0                   ; used by QuickDraw GX

   JMP      BWInitialize        ; override for GXInitialize
   JMP      BWShutDown          ; override for GXShutDown
   JMP      BWJobPrintDialog      ; override for GXJobPrintDialog
   JMP      BWHandlePanelEvent    ; override for GXHandlePanelEvent
   JMP      BWCreateSpoolFile     ; override for GXCreateSpoolFile
   JMP      BWDespoolPage         ; override for GXDespoolPage
   JMP      BWCloseSpoolFile      ; override for GXCloseSpoolFile

   ENDPROC

END
```

Note

The code shown in Listing 2-2 is for the MPW environment. If you are programming in a different development environment, you might need to use different assembler directives. You must, however, be certain to include the initial 4 bytes (set to 0) and each `JMP` statement. u

The `EXPORT` statement at the beginning makes the jump table public. The `IMPORT` statements make it possible for your assembly-language program to reference the C language functions performing your message overrides and to provide correct linkage to those functions.

The name of each override function provided by the background picture printing extension is prefixed with `BW` to differentiate it from other overrides of the same message. This means that the extension's override of the `GXInitialize` message is named `BWInitialize`, its override of the `GXDespoolPage` message is named `BWDespoolPage`, and so on.

Because QuickDraw GX uses the first 4 bytes in the jump table to store the owner count value, the first statement (`DC.L`) must be included. These bytes must all have the value 0 in them.

You must include one `JMP` statement for each message that you override. You can choose to intersperse the `IMPORT` and `JMP` statements as shown in Listing 3-2 on page 3-15 in the chapter "Printer Drivers," or you can place all of the `IMPORT` statements together, followed by all of the `JMP` statements, as is shown in Listing 2-2.

IMPORTANT

Always coordinate the entries in your override resources with the entries in your jump table. If they are not aligned, the wrong code will be executed to override a message. The offset that you specify in the resource for each message must match the offset of the corresponding override function in your jump table. You must also include 4 bytes with zero values at the beginning of your jump table. s

The Printing Message Overrides

The code that makes up your printing extension is a set of functions that override, either partially or totally, some of the printing messages that QuickDraw GX sends during the process of printing a document. The contents of the file `backwash.c`, which contains the source code for the message overrides in the background picture printing extension, are shown in the QuickDraw GX sample code.

QuickDraw GX provides a default implementation of each printing message that it sends. You can augment (partially override) some printing messages, and you can replace (totally override) others. For each printing message, QuickDraw GX provides one of three kinds of default implementations, as shown in Table 2-3.

Table 2-3 Implementation types for QuickDraw GX printing messages

Message type	**Default implementation**
Empty	Does nothing, so you can provide an override of this message to insert functionality into the printing process. You can choose to partially or totally override this message.
Can be partially overridden	Provides necessary functionality, which you augment by partially overriding it. You must forward this message to other message handlers.
Can be totally overridden	Provides basic functionality, which you can either partially or totally override.

Whenever you override a QuickDraw GX printing message, you must be certain that the declaration of your override function matches the declaration of the message. This means that the type of function return and the type of each parameter must match the types in the message declaration. The chapter “Printing Messages” shows the declaration of each printing messages.

When you partially override a printing message, you add some functionality to that provided by other message handlers, including QuickDraw GX (through its default implementation), the printer driver, and other printing extensions. You forward the message to these other handlers, as described in the section “Forwarding Messages” on page 2-13.

When you totally override a message, you replace any functionality that is provided by other message handlers, including the default implementation. Your total message override does not forward the message to the other handlers. This means that you must be sure to replace the functionality that is provided by the default implementation.

Although the default implementation of a message might be empty, other printing message handlers (the printer driver and other printing extensions) can also override messages, so you need to carefully consider whether or not to create a total override of a message.

Whether or not you can totally override a message and when you need to forward a message in your implementation of a partial override is specific to each message. The reference section “Printing Messages Reference” beginning on page 4-9 in the chapter “Printing Messages” provides this information for each message.

Choosing the Messages to Override

Which messages you need to override for your printing extension depends entirely on what you want your extension to do. There are not any messages that you are required to override. Some extensions override many messages to provide their operations, and many extensions are created by overriding only a few messages. Refer to the chapter "Printing Messages" in this book for complete information about the available messages.

Forwarding Messages

Your printing extension can forward a message in its implementation of a partial override. If you are totally overriding a message, you do not forward it to other message handlers. You can forward the message either before or after performing your own actions. To forward a message to the next message handler in the message chain, use a statement with the following format:

```
anErr = Forward_MessageName(arguments);
```

For example, the background picture printing extension overrides the `GXJobPrintDialog` message to add a panel to the Print dialog box. This message must be forwarded so that the default implementation can build the default dialog box and any other message handlers can add their panels to the dialog box. Listing 2-3 shows the override of the `GXJobPrintDialog` message from the background picture extension.

Listing 2-3 Forwarding the `GXJobPrintDialog` message

```
OSErr BWJobPrintDialog (gxDialogResult *dlogResult)
{
   OSErr err;

   err = SetupPrintPanel();
   if (!err)
      err = Forward_GXJobPrintDialog(dlogResult);
   return err;
}
```

This override function calls a local function named `SetupPrintPanel` to add items to the Print dialog box that are used for controlling the background picture extension. If that function succeeds, this version forwards the message to the next message handler, which can add its own panel. The `GXJobPrintDialog` message is described on page 4-84 in the chapter "Printing Messages."

Sending Messages

Your printing extension can also send a printing message to other handlers in the message chain. When you send a message, QuickDraw GX receives it and then sends it to the first message handler in the chain. To send a message, use a statement with this format:

```
anErr = Send_GXMessageName(arguments);
```

For example, to send the `GXBufferData` message, which is described on page 4-139 in the chapter "Printing Messages," use the following statement:

```
anErr = Send_GXBufferData(gxDialogResult *);
```

For more information on sending messages, refer to the chapter "Message Manager" in *Inside Macintosh: QuickDraw GX Environment and Utilities.*

Handling Exceptions in Your Message Overrides

The code samples presented in this chapter make use of an exception-handling strategy that simplifies the job of testing for error conditions after each function call. This strategy uses three C macros, each of which branches to an error-handling statement in response to a condition.

The first macro, `nrequire`, branches to a label when the value of its `condition` argument is anything other than 0. This macro is commonly used to test the result code from a function call and branch if the function returned an error (any value other than `noErr`, which is 0). The syntax of the `nrequire` macro is

```
   nrequire(condition, location);
```

An example of using the `nrequire` macro is shown in Listing 2-4.

Listing 2-4 Using the `nrequire` macro for exception handling

```
   OSErr        err;
   long         count;
   short        pictRefNum;
   PicHandle    backwashPict = nil;

   err = FSpOpenDF(opFSSpec, fsCurPerm, &pictRefNum);
   nrequire(err, CouldNotOpenFile);

   err = GetEOF(pictRefNum, &count);
   nrequire(err, CouldNotGetEOF);
   count -= 512;
   err = SetFPos(pictRefNum, fsFromStart, 512);
   nrequire(err, CouldNotSetFilePos);

   ...


CouldNotGetEof:
CouldNotSetFilePos:
   FSClose(pictRefNum);

CouldNotOpenFile:
   return(backwashPict);
}
```

The function in Listing 2-4 uses the `nrequire` macro instead of using an `if` statement to test the value of `err` after each call. The `nrequire` macro performs a branch to the specified label if the value of `err` is anything other than `noErr`. Each label that is referenced by the `nrequire` macro leads to a call that cleans up after the error.

A variation of the `nrequire` macro that is used in some functions is the `require` macro. This macro is exactly the same as `nrequire` except that it branches when its `condition` argument is 0 (whereas `nrequire` branches on anything other than 0). The syntax of the `require` macro is

```
   require(condition, location);
```

A final variation of the `nrequire` macro that is used in some of the functions of the background picture printing extension is the `nrequire_action` macro. This macro works the same way as `nrequire` and additionally performs an action before branching. The syntax of the `nrequire_action` macro is

```
   nrequire_action(condition, location, action);
```

The action is performed only if the value of the `condition` argument is anything other than 0. The action is performed before branching to the specified label. An example of the `nrequire_action` macro is shown in Listing 2-5.

Listing 2-5 Using the `nrequire_action` macro for exception handling

```
   grErr = GetJobCollectionItem(&backwashConfig, nil,
                  kBackwashCollectionType, kBackwashSettingsID);
   nrequire_action((!grErr && !backwashConfig.addBackwash &&
                                    backwashConfig.haveFileInfo),
                   NotAddingBackwash, grErr = noErr);
```

The `nrequire_action` call in this example assigns the value `noErr` to the `grErr` variable before branching to the `NotAddingBackwash` label.

Implementing the Background Picture Extension Functions

This section describes the functions of the background picture printing extension. This extension needs to accomplish the following tasks:

1. Initialize the environment.
2. Add a panel to the Print dialog box so that the user can make several choices: whether to enable or disable this extension, which picture to use as the background picture, and at what intensity level to draw the picture on the page.
3. Respond to events that occur in the panel, such as a mouse click or keypress.
4. Store the background picture in the spool file along with the document that is being spooled.
5. Draw the selected background picture on each page of a document that is being despooled.
6. Deallocate the storage allocated for the background picture shape that was spooled.
7. Shut down the environment at the end of the program.

Initializing the Extension Environment

The background picture printing extension performs very simple initialization. Its override of the `GXInitialize` message, `BWInitialize`, allocates a globals world so that it has the global data needed for a printing extension. The `GXInitialize` message is described on page 4-43 in the chapter "Printing Messages." The code for the `BWInitialize` function is shown in Listing 2-6.

Listing 2-6 The `BWInitialize` override function

```
OSErr BWInitialize()
{
   OSErr err = noErr;

   err = NewMessageGlobals(A5Size(), A5Init);
   if (!err) err = InitGlobalData();
   return err;
}
```

The `BWInitialize` function first sets up an A5 world, which you must do in your extension if you are going to use global data. Setting up an A5 world means setting the A5 register to reference your global data. The functions `A5Size` and `A5Init` are supplied for creating an A5 world.

`BWInitialize` calls a local function named `InitGlobalData` to initialize any global variables that are defined by the background picture printing extension. This function is shown in Listing 2-7.

IMPORTANT
You must perform the actual initialization of your global variables in a function that you call from your override of the `GXInitialize` message. This is because some compilers optimize code in a way that can invalidate the A5 register. This is why the `BWInitialize` override function calls the `InitGlobalData` function to initialize the globals. s

Listing 2-7 The `InitGlobalData` function

```
OSErr InitGlobalData()
{
   gBackwashShape = nil;
   return noErr;
}
```

This function initializes the `gBackwashShape` variable to `nil` to indicate that it is currently unused.

Adding a Background Picture Panel to the Print Dialog Box

The background picture printing extension adds a panel to the Print dialog box. The placeholders for this panel are defined in the `backwash.r` resource file; however, several of the values need to be filled in at run time. In the background picture printing extension, the override of the `GXJobPrintDialog` message, `BWJobPrintDialog`, performs this operation by calling a local function, `SetupPrintPanel`, and then forwarding the `GXJobPrintDialog` message to the rest of the message handlers. The `GXJobPrintDialog` message is described on page 4-84 in the chapter “Printing Messages.” The `BWJobPrintDialog` function is shown in Listing 2-8.

Listing 2-8 The `BWJobPrintDialog` override function

```
OSErr BWJobPrintDialog(gxDialogResult *dlogResult)
{
   OSErr err;

   err = SetUpPrintPanel();

   if (!err)
      err = Forward_GXJobPrintDialog(dlogResult);

   return err;
}
```

The local function `SetupPrintPanel` performs the work of setting up the panel that the background picture extension adds to the Print dialog box. It uses the Collection Manager to store and access a `BackwashCollection` structure. The `BackwashCollection` structure is defined as shown in Listing 2-9. The Collection Manager is described in *Inside Macintosh: QuickDraw GX Environment and Utilities*.

Listing 2-9 The `BackwashCollection` structure

```
struct BackwashCollection {
   long           intensity;
   unsigned char  addBackwash;
   Boolean        haveFileInfo;
   FSSpec         fileInfo;
};

typedef struct BackwashCollection BackwashCollection;
```

The fields of this structure are used as follows:

Field descriptions

`intensity`	The intensity level (a percentage value) that the user has selected for drawing the background picture on the page.
`addBackwash`	A value that indicates whether the user has enabled the background picture printing extension.
`haveFileInfo`	A Boolean value that indicates whether the user has entered a picture filename into the panel.
`fileInfo`	The file system specification of the picture file that the user chose to use as the background picture.

The `SetupPrintPanel` function first attempts to access the background picture collection item from the job collection, which is the collection of items maintained by the Collection Manager for the print job that is printing. If this is the first time that the background picture extension has been accessed, the item is not found, and `SetupPrintPanel` creates a new item with default values and adds it to the collection. Finally, `SetupPrintPanel` sets up the panel information for the Dialog Manager and calls the `GXSetupDialogPanel` function to actually add the panel to the dialog box. The `GXSetupDialogPanel` function is described on page 5-28 in the chapter “Printing Functions for Message Overrides.” The code for the `SetupPrintPanel` function is shown in Listing 2-10.

Listing 2-10 The `SetupPrintPanel` function

```
OSErr SetUpPrintPanel()
{
   OSErr                noErr;
   gxPanelSetupRecord   panelSetupRec;
   BackwashCollection   backwashConfig;

/*
   Get the job collection and then try to find the
   backwash collection item in there.
*/

   jobCollection = GXGetJobCollection(GXGetJob());
   /* call local function to get settings */
   err = GetJobCollectionItem(&backwashConfig, nil,
               kBackwashCollectionType, kBackwashSettingsID);

/*
   If the collection item doesn't yet exist, create a new
   item, set it up with default values, and add it to the
   job collection
*/

   if (err == collectionItemNotFoundErr)
      {
         backwashConfig.addBackwash = kDontAddBackwash;
         backwashConfig.haveFileInfo = false;
         backwashConfig.intensity = kDefaultIntensity;
         strcpy(&backwashConfig.fileInfo.name[1], "none");
         backwashConfig.fileInfo.name[0] =
               strlen(&backwashConfig.fileInfo.name[1]);

         /* call local function to store settings */
         err = StoreJobCollectionItem(&backwashConfig,
                                    sizeof(BackwashCollection),
                                    kBackwashCollectionType,
                                    kBackwashSettingsID, true);
      }

   nrequire(err, GetSettings_Failed);

/*
   Set up the panel: store the ID of the panel resource to use,
   the resource file in which it is located, and the type of
   panel that is being stored.
*/
   panelSetupRec.panelResId= r_BackwashPanel;   /* resource ID */
   /* resource file */
   panelSetupRec.resourceRefNum = GXGetMessageHandlerResFile();
   panelSetupRec.refCon = 0;              /* not used*/
   panelSetupRec.panelKind = gxExtensionPanel;  /* panel type */

   err = GXSetupDialogPanel(&panelSetupRec);

GetSettings_Failed:

   return err;
}
```

SetupPrintPanel returns an error code of type OSErr, which is the same type of function result that is returned by all of the functions that it calls. SetupPrintPanel also uses the error code to detect when it needs to initially add its own item to the job collection.

s **WARNING**
The size of the items in the job collection is subject to change as QuickDraw GX evolves. For that reason, it is important for you to specify an expected size for each item in your calls to the Collection Manager, rather than specifying nil, which tells the Collection Manager to copy the entire object no matter what its size is. The size that you specify must match the size of the data structure into which the item is being copied. s

Handling an Event in the Background Picture Panel

When a user event occurs in the background picture panel, the background picture printing extension needs to examine the event and determine whether or not to handle it. QuickDraw GX sends the GXHandlePanelEvent message whenever an event occurs in a printing panel. The GXHandlePanelEvent message is described on page 4-85 in the chapter “Printing Messages.” Included with the message is a structure of type gxPanelInfoRecord, which is declared as shown in Listing 2-11.

Listing 2-11 The gxPanelInfoRecord structure

```
struct gxPanelInfoRecord {
   gxPanelEvent panelEvt;  /* the event */
   short panelResId;       /* resource ID of current 'ppnl' res */
   DialogPtr pDlg;         /* pointer to dialog box */
   EventRecord *theEvent;  /* pointer to event */
   short itemHit;          /* actual item number of event */
   short itemCount;        /* number of items before your items */
   short evtAction;        /* the action that occurs after
```

```
                               this event is processed */
   short errorStringId;    /* ID of error string */
   gxFormat theFormat;     /* the current format */
   void *refCon;           /* refCon from gxPanelSetupRecord */
};

typedef struct gxPanelInfoRecord gxPanelInfoRecord;
```

The fields of this structure are described in the section “The Panel Information Structure” on page 4-35 in the chapter “Printing Messages.”

Panel events and the way that they are processed are described in *Inside Macintosh: QuickDraw GX Printing*.

In the background picture printing extension, the override of the `GXHandlePanelEvent` message, `BWHandlePanelEvent`, responds to the panel events that are shown inTable 2-4. The list of possible panel events that you can respond to is given in the section “Panel Events” on page 4-36 in the chapter “Printing Messages.”

Table 2-4 Panel events handled by the background picture printing extension

Constant	Explanation
`gxPanelOpenEvt`	The panel is about to open. It needs to be initialized and drawn. The background picture extension responds to this event by initializing the current filename that is displayed in the panel.
`gxPanelHitEvt`	The user has selected an item in the panel. The background picture extension responds when the user selects the “Select PICT” panel item by replacing the file specification in the job collection with the filename that the user selects.
`gxPanelActivateEvt`	The dialog box window in which the panel resides has just been activated. The background picture extension responds to this event by activating the picture intensity field.
`gxPanelDeactivateEvt`	The dialog box window in which the panel resides is about to be deactivated. The background picture extension responds to this event by deactivating the picture intensity field.
`gxPanelIconFocusEvt`	The focus has changed from the panel to the icon list. The background picture extension responds to this event by deactivating the picture intensity field.
`gxPanelPanelFocusEvt`	The focus has changed from the icon list to the panel. The background picture extension responds to this event by activating the picture intensity field.

The `BWHandlePanelEvent` function, which is shown in Listing 2-12, operates as a `switch` statement that selects the events of interest (those listed in Table 2-4) and provides code to handle each. The `gxPanelOpenEvt` case is handled by calling a local function, `OpenBackwashPanel`, which is shown in Listing 2-13 on page 2-24.

Listing 2-12 The `BWHandlePanelEvent` override function

```
OSErr BWHandlePanelEvent(gxPanelInfoRecord *panelInfo)
{
   OSErr                err = noErr;
   GrafPtr              oldPort;
   DialogPtr            pDlg;
   StandardFileReply    reply;
   SFTypeList           typeList;
   BackwashCollection   backwashConfig;
   Handle               hItem;
   Rect                 itemRect;
   short                itemType;

   pDlg = panelInfo->pDlg;
   GetPort(&oldPort);
   SetPort(pDlg);

   switch (panelInfo->panelEvt)  /* select events of interest */
   {
      /* if the panel is opening, initialize it */
      case gxPanelOpenEvt:
         OpenBackwashPanel(pDlg, panelInfo->itemCount);
         break;

/*
   If the user clicks the Select Picture button, prompt for the
   name of a file to load. If the user selects one, access the
   collection item, move the user’s file specification to the
   collection, and replace the old collection item with the
   modified version.
*/
      case gxPanelHitEvt:
         if (panelInfo->itemHit == (panelInfo->itemCount +
                                    d_SelectPicture))
         {
            typeList[0] = 'PICT';
            StandardGetFile(nil, 1, typeList, &reply);
            require(replay.sfGood, UserCancelledFileDialog);

            err = GetJobCollectionItem(&backwashConfig, nil,
                  kBackwashCollectionType, kBackwashSettingsID);

            nrequire(err, GetSettings_Failed);

            backwashConfig.haveFileInfo = true;
            BlockMove(&reply.sfFile,
                      &backwashConfig.fileInfo, sizeof(FSSpec));

            /* replace old collection item with updated version */
            err = StoreJobCollectionItem(&backwashConfig,
                        sizeof(BackwashCollection),
                        kBackwashCollectionType,
                        kBackwashSettingsID, false);
            nrequire(err, StoreSettings_Failed);

            /* update the filename dialog item */
            GetDItem(panelInfo->pDlg,
                     panelInfo->itemCount +d_FileNameItem,
                     &itemType, &hItem, &itemRect);
            SetIText(hItem, &backwashConfig.fileInfo.name);
         }
         break;

/*
   If the panel is activating or deactivating, or if the focus
   (the section of the dialog box that is active) is changed,
   either activate or deactivate the picture intensity field.
*/
      case gxPanelActivateEvt:
      case gxPanelDeactivateEvt:
      case gxPanelIconFocusEvt:
      case gxPanelPanelFocusEvt:
         if ((((DialogPeek) pDlg)->editField +1) ==
                          (panelInfo->itemCount +d_intensity))
         {
            if ((panelInfo->panelEvt == gxPanelPanelFocusEvt)
               TEActivate(((DialogPeek) pDlg)->textH);
            else
               TEDeactivate(((DialogPeek) pDlg)->textH);
         }
         break;
   }

UserCancelledFileDialog:
GetSettings_Failed:
StoreSettings_Failed:
   SetPort(oldPort);
   return err;
}
```

The `OpenBackwashPanel` function in Listing 2-13 is a local function called by the `BWHandlePanelEvent` function to initialize the background picture panel when the user opens it. This function performs any initialization of the panel that cannot be performed by `'xdtl'` resource specifications, which are described in *Inside Macintosh: QuickDraw GX Printing*.

Listing 2-13 The `OpenBackwashPanel` function

```
/*
   OpenBackwashPanel handles non-'xdtl' item initialization when
   the panel is opened. Note that the items are offset from the
   the value of itemCount, which means that item #5 in the panel
   is accessed by passing the value itemCount+5.
*/

void OpenBackwashPanel(DialogPtr pDlg, short itemCount)
{
   BackwashCollection   backwashConfig;
   Handle               hItem;
   Rect                 itemRect;
   short                itemType;

/*
   Initialize the current filename displayed, based on the
   settings in the backwash collection item.
*/

   GetJobCollectionItem(&backwashConfig, nil,
               kBackwashCollectionType, kBackwashSettingsID);

   GetDItem(pDlg, itemCount +d_FileNameItem, &itemType,
            &hItem, &itemRect);
   SetIText(hItem, &backwashConfig.fileInfo.name);
}
```

The OpenBackwashPanel function initializes the dialog box panel by filling in the filename that is stored in its configuration information. It operates by retrieving the configuration information from the job collection and storing the filename from the configuration into the filename item in the panel.

Storing the Background Picture in the Spool File

The background picture printing extension draws the background picture on each page. To do this, it first stores the background picture as a resource in the spool file. It then accesses that resource and applies the picture to each page as the page is being despooled.

To store the picture in the spool file, the background picture printing extension overrides the GXCreateSpoolFile message, which QuickDraw GX sends at the start of spooling a document. The GXCreateSpoolFile message is described on page 4-68 in the chapter "Printing Messages." In the background picture printing extension, the override of GXCreateSpoolFile, BWCreateSpoolFile, is shown in Listing 2-14.

Listing 2-14 The BWCreateSpoolFile override function

```
OSErr BWCreateSpoolFile(FSSpecPtr anFSSpec, long createOptions,
                        gxSpoolFile *theSpoolFile)
{
   gxGraphicsError      grErr;
   BackwashCollection   backwashConfig;

   /* forward the message so that the spool file is created */
   grErr = (gxGraphicsError) Forward_GXCreateSpoolFile(anFSSpec,
                                    createOptions, theSpoolFile);
   nrequire(grErr, ForwardMessage_Failed);
/*
   Get the collection item and see if extension is enabled and
   picture file info has been entered. If so, add the background
   picture to the spool file.
*/

   grErr = (gxGraphicsError) GetJobCollectionItem(&backwashConfig,
               nil, kBackwashCollectionType, kBackwashSettingsID);

   if (!grErr && backwashConfig.addBackwash &&
                                    backwashConfig.haveFileInfo)
```

```
      grErr = AddBackwash(&backwashConfig.fileInfo,
               (short) backwashConfig.intensity, *theSpoolFile);
   else
      if (grErr == collectionItemNotFoundErr)
         grErr = noErr;

ForwardMessage_Failed:
   return (OSErr) grErr;
}
```

The `BWCreateSpoolFile` function first forwards the `GXCreateSpoolFile` message to allow QuickDraw GX to use the default implementation (or other message handlers to use their implementations) to create the spool file. `BWCreateSpoolFile` then calls the local function `AddBackwash` to read the background picture file and store it in the spool file. The `AddBackwash` function is shown in Listing 2-15.

Listing 2-15 The AddBackwash function

```
gxGraphicsError AddBackwash(FSSpecPtr fileInfo,
                           short theIntensity,
                           gxSpoolFile theSpoolFile)
{
   gxGraphicsError   grErr = noErr;
   Rect              pictBounds;
   PicHandle         backwashPict;
                     /* use {1, 1} to mean don't stretch */
   Point             patStretchPoint = {1,1};
   Handle            gxShapeHdl = nil;
   gxShape           gxPictShape;
   short             resAttribs;

   /* load the picture and then convert it to a GX shape */
   backwashPict = LoadAPict(fileInfo);
   require_action(backwashPict, CouldNotLoadPICT,
                                          grErr = nilHandleErr;);

   pictBounds = (*backwashPict)->picFrame;
   gxPictShape = GXNewShape(gxPictureType);  /* picture shape */
   nrequire_action(GXGetGraphicsError(&grErr),
               CouldNotCreateShape, KillPicture(backwashPict););

   GXConvertPICTToShape(backwashPict, gxDefaultOptionsTranslation,
                   &pictBounds, &pictBounds, patStretchPoint,
                   gxPictShape, nil);
   KillPicture(backwashPict);       /* get rid of original */
   nrequire_action(GXGetGraphicsError(&grErr),
            CouldNotConvertShape, GXDisposeShape(gxPictShape););

   /* lighten the picture shape as specified by user */
   grErr = SetShapeIntensity(gxPictShape, theIntensity);
   nrequire_action(grErr, CouldNotSetShapeIntensity,
                                 GXDisposeShape(gxPictShape););

   /* flatten picture shape into handle */
   gxShapeHdl = ShapeToHandle(gxPictShape);
   grErr = (gxGraphicsError) MemError();
   GXDisposeShape(gxPictShape);
   nrequire(grErr, CouldNotFlattenShape);
/*
   Add the picture shape as a resource in the spool file. After
   adding it, mark the resource as “sysHeap, purgeable”, so that
   it won’t be loaded into PrinterShare’s heap, which is small.
*/
   grErr = Send_GXSpoolResource(theSpoolFile, gxShapeHdl,
                  kBackwashCollectionType, r_BackwashPICTID);
   nrequire_action(grErr, CouldNotAddShape,
                                    DisposHandle(gxShapeHdl););

   resAttribs = GetResAttrs(gxShapeHdl);
   SetResAttrs(gxShapeHdl,
                     resAttribs | resSysHeap | resPurgeable);
   ChangedResource(gxShapeHdl);
   WriteResource(gxShapeHdl);

   /* release the resource–DO NOT call DisposHandle on it */
   ReleaseResource(gxShapeHdl);

CouldNotLoadPICT:
CouldNotCreateShape:
CouldNotConvertShape:
CouldNotSetShapeIntensity:
CouldNotFlattenShape:
CouldNotAddShape:

   return grErr;
}
```

The AddBackwash function stores the background picture as a resource in the spool file. AddBackwash first calls a local function, LoadAPict, to read a picture file and create a PicHandle for it. The LoadAPict function, which opens the specified file, creates the handle, skips over the file header information, and reads the picture data into the handle. AddBackwash then converts the picture into a QuickDraw GX shape by calling the GXConvertPICTToShape function, which is described in *Inside Macintosh: QuickDraw GX Objects*.

Once AddBackwash has a shape object that represents the background picture, it calls a local function, SetShapeIntensity, to lighten the background picture to the intensity level selected by the user (in the background picture panel). SetShapeIntensity modifies the transfer object for each shape in the background picture to use blending, which lightens the final picture. The intensity is adjusted to match the value selected by the user.

Finally, AddBackwash flattens the picture shape into a handle by calling the local function ShapeToHandle and adds the flattened shape to the spool file as a resource. The resource is added in the system heap and is marked as purgeable. ShapeToHandle calls the GXFlattenShape function, which is described in *Inside Macintosh: QuickDraw GX Objects*.

Adding the Background Picture to a Page

The background picture printing extension draws the background picture on each page as the page is being despooled. It does this by overriding the GXDespoolPage message with the BWDespoolPage function, which is shown in Listing 2-16. This function first forwards the GXDespoolPage message, which is described on page 4-75 in the chapter “Printing Messages,” so that any other message handlers that are going to modify the page can do so first, which allows the background picture to be drawn using the final shape of the page.

Listing 2-16 The BWDespoolPage function

```
OSErr BWDespoolPage(gxSpoolFile theSpoolFile, long thePageNum,
    gxFormat theFormat, gxShape *thePage, Boolean *formatChanged)
{
   OSErr                anOSErr;
   gxGraphicsError      grErr;
   Handle               vpListHdl, gxShapeHdl = nil;
   gxRectangle          pBounds;
   Fixed                cx, cy, px, py;
   long                 numViewPorts;
   BackwashCollection   backwashConfig;

/*
   First, forward the message to make sure that the final page
   shape is being despooled.
*/

   anOSErr = Forward_GXDespoolPage (theSpoolFile,
                  thePageNum, theFormat, thePage, formatChanged);
   nrequire((grErr = (gxGraphicsError) anOSErr),
                      ForwardMessage_Failed);

   grErr = GetJobCollectionItem(&backwashConfig, nil,
                  kBackwashCollectionType, kBackwashSettingsID);
   nrequire_action((!grErr && backwashConfig.addBackwash
                     && backwashConfig.haveFileInfo),
                   NotAddingBackwash, grErr = noErr);
/*
   Load the flattened shape resource (the background picture) if
   necessary. This is only needed for the first page because the
   shape reference is stored in a global variable.
*/
   if (gBackwashShape == nil)
      anOsErr = Send_GXDespoolResource(theSpoolFile,
         kBackwashCollectionType, r_BackwasPICTID, &gxShapeHdl);
   nrequire((grErr = (gxGraphicsError) anOSErr),
            DespoolResource_Failed);
/*
   If necessary, unflatten the background picture shape. Since a
   view port list is needed for the unflattening, use the current
   page’s view port list.
*/
   if (gBackwashShape == nil)    /* first page */
   {
      numViewPorts = GXGetShapeViewPorts(*thePage, nil);
      vpListHdl = TempNewHandle(numViewPorts * sizeof(gxViewPort),
                                 &anOSErr);
      grErr = (gxGraphicsError) anOSErr;
      nrequire_action(grErr, TempNewHandle_Failed,
                        ReleaseResource(gxShapeHdl););
      HLock(vpListHdl);
      GXGetShapeViewPorts(*thePage, (gxViewPort *) *vpListHdl);
      gBackwashShape = HandleToShape(gxShapeHdl, numViewPorts,
                                     (gxViewPort *) *vpListHdl);
      DisposHandle(vpListHdl);/* all done with this */

      if (gxShapeHdl)
```

```
            ReleaseResource(gxShapeHdl);
        nrequire(GXGetGraphicsError(&grErr), CouldNotSetUpShape);
    }
/*
    Now use the current page format to obtain the page dimensions.
    Use the dimensions and the bounds of the picture shape to
    center the page. Once the shape is centered, add it behind the
    shapes that constitute the contents of the page.
*/
    GXGetFormatDimensions(theFormat, &pBounds, nil);
    grErr = GXGetJobError(GXGetJob());
    nrequire(grErr, GetFormatDims_Failed);

    cx = pBounds.left + (pBounds.right - pBounds.left) >>1;
    cy = pBounds.top + (pBounds.bottom - pBounds.top) >>1;

    GXGetShapeBounds(gBackwashShape, 0, &pBounds);
    px = pBounds.left + (pBounds.right - pBounds.left) >>1;
    py = pBounds.top + (pBounds.bottom - pBounds.top) >>1;
    GXMoveShapeTo(gBackwashShape, cx-px, cy-py);

    GXSetPictureParts(*thePage, 1, 0, 1, &gBackwashShape,
                                                nil, nil, nil);
    GXGetGraphicsError(&grErr);

ForwardMessage_Failed:
NotAddingBackwash:
DespoolResource_Failed:
TempNewHandle_Failed:
CouldNotSetUpShape:
GetFormatDims_Failed:

    return grErr;
}
```

After forwarding the `GXDespoolPage` message, `BWDespoolPage` makes sure that the background-picture-configuration collection item is available in the job collection. If the item is not found in the job collection, it means that the background picture panel was never opened because that event triggers the adding of the item to the collection. If the panel was never opened, then the user is printing without dialog boxes, and no background picture is added to the pages. In this case, `BWDespoolPage` does not return an error because the function did not really fail. This is accomplished in the `nrequire_action` call, which changes the error code to `noErr` and branches to the end of the function.

If the background-picture-configuration collection item is found, BWDespoolPage checks to see if the global variable for the background picture shape is nil. If so, BWDespoolPage reads the resource from the spool file and unflattens it into a picture shape.

Finally, BWDespoolPage centers the background picture shape on the page and adds the shape to the page contents.

Closing the Spool File

When QuickDraw GX finishes with the spooling of a document, it sends the GXCloseSpoolFile message, which is described on page 4-79 in the chapter "Printing Messages." The background picture printing extension overrides this message with the BWCloseSpoolFile function, shown in Listing 2-17, to dispose of the picture shape that it allocated for the background picture.

Listing 2-17 The BWCloseSpoolFile override function

```
OSErr BWCloseSpoolFile(gxSpoolFile theSpoolFile,
                        long closeOptions)
{
   if (gBackwashShape != nil)
   {
      GXDisposeShape(gBackwashShape);
      gBackwashShape = nil;
   }
   return Forward_GXCloseSpoolFile(theSpoolFile, closeOptions);
}
```

Shutting Down the Background Picture Extension Environment

When the background picture printing extension is finished, it needs to deallocate the globals that it allocated in its override of the GXInitialize message. In the background picture extension, the override of the GXShutDown message, BWShutDown, calls the DisposeMessageGlobals function to do that deallocation, as shown in Listing 2-18. The GXShutDown message is described on page 4-44 in the chapter "Printing Messages."

Listing 2-18 The BWShutDown override function

```
OSErr BWShutDown()
{
   DisposeMessageGlobals();
   return noErr;
}
```

Using Resources in Printing Extensions

You must create a number of resources that define your printing extension and provide QuickDraw GX with the information that it needs to properly load and execute the extension. The resources an extension contains are defined in a printing extension file of type `'pext'`. Each printing extension file must include certain resources and may include some optional resources, as shown in Table 2-5.

Table 2-5 Resource types used to define a printing extension

Resource type	Count	Description
`'vers'`	1 or more	Records version and compatibility information for your extension
`'scop'`	1 or more	Defines with which devices and drivers the extension is compatible
`'eopt'`	1	Tells QuickDraw GX when to load the extension
`'over'`	1-4	Defines which messages your extension needs to receive
`'load'`	1	Specifies the order in which extensions are installed in the message chain
`'stat'`	0 or more	Defines status messages for display
`'plrt'`	0 or more	Defines printing alert messages for display
`'ppnl'`	0 or more	Defines the contents of a panel that you are adding to a dialog box
`'xdtl'`	0 or more	Provides information that QuickDraw GX needs to execute panel controls
`'ptyp'`	0 or more	Defines characteristics of a paper type

Most of these resource types are described in the chapter “Printing Resources” in this book. The panel (`'ppnl'`), extended panel (`'xdtl'`), and paper type (`'ptyp'`) resources are described in *Inside Macintosh: QuickDraw GX Printing*. This section provides examples of these resources as used in the background picture printing extension. The contents of the file `backwash.r`, which contains the resource definitions for the background picture printing extension, are shown in the QuickDraw GX sample code.

To avoid conflicts with resources defined by printer drivers, application programs, and Macintosh system software, the resources that you define for your printing extension must have IDs in the range 0x9600 (-27136) through 0x97FF (-26625). Most of the resources have specific ID values that you must use with them; these IDs are shown in the examples in this chapter and are defined in the resource descriptions in the chapter “Printing Resources” in this book.

All of the resources that you define for your printing extensions need to be loaded into the system heap and need to be purgeable. System resources are stored in the system heap as opposed to the application heap, where application resources are stored. Purgeable resources can be purged by the Memory Manager when space is required, as described in *Inside Macintosh: Memory*. You need to specify these attributes in the first line of every resource that you define for your extensions, as is done in every resource example in this chapter.

Defining Code Segments in Your Printing Extension

The code segments that you use to create your printing extension must use segment type `'pext'`, which is defined by the constant `gxPrintingExtensionType`. The ID of your first code segment needs to be 0, and you need to increment the ID by 1 for each subsequent segment in your extension.

Defining Version Compatibility for Your Printing Extension

Your printing extension must contain at least one version (`'vers'`) resource that defines its compatibility with QuickDraw GX. Version resources are used to record version information for Macintosh applications. You need to include a version resource with an ID of `gxPrintingExtensionBaseID` that defines with which version of QuickDraw GX your extension is compatible.

IMPORTANT

You must include the version resource or your extension will not load. s

For the current version, the value of the first byte of the resource definition must be either 1 or 0. Listing 2-19 shows the version resource that defines QuickDraw GX compatibility for the background picture printing extension.

Listing 2-19 The QuickDraw GX version resource for the background picture printing extension

```
resource 'vers' (gxPrintingExtensionBaseID, sysHeap, purgeable) {
   0x0,
   0x0,
   release,
   0x0,
   verUS,
   "",
   ""
};
```

You can also include standard version resources in the resource files for your printing extension. These resources are described in *Inside Macintosh: Macintosh Toolbox Essentials*. Listing 2-20 shows the standard version resources for the background picture printing extension.

Listing 2-20 The standard version resources for the background picture printing extension

```
resource 'vers' (1, sysHeap, purgeable) {
   0x1,
   0x0,
   beta,
   0x2,
   verUS,
   "1.0b2",
   "1.0b2, © Apple Computer, Inc.  1992-1993"
};

resource 'vers' (2, sysHeap, purgeable) {
   0x1,
   0x0,
   beta,
   0x2,
   verUS,
   "1.0b2",
   "Backwash v1.0b2"
};
```

Defining the Scope of Your Printing Extension

When you write a printing extension, you must define the types of devices and drivers with which your extension is compatible. You do this by including an extension scope (`'scop'`) resource in your extension. The extension scope resource is described in detail on page 6-26 in the chapter “Printing Resources.”

The scope of your printing extension can range from very general to quite specific. The scope of some extensions is known as universal scope, which means that the extension can run on any device. The scope of other extensions is defined in terms of the specific devices on which the extension can run. And the scope of yet other extensions is defined in terms of the specific devices on which the extension cannot run. Many extensions that affect only the final image on the printed page can be defined with universal scope.

There are three kinds of extension scope resources used in printing extensions, each of which has a unique ID defined for it. The constants for these IDs are shown in Table 2-6.

Table 2-6 Identifiers for the extension scope resource

Constant	Explanation
`gxDriverScopeID`	Defines with which imaging systems the extension is compatible. Four imaging systems types are supported: raster, PostScript, vector, and universal.
`gxPrinterScopeID`	Defines specific devices with which the extension is compatible.
`gxPrinterExceptionScopeID`	Defines specific devices with which the extension is not compatible.

The declarations of these and other constants that are used in printing extension resources are shown in the section "Summary of Printing Resources" beginning on page 6-90 in the chapter "Printing Resources."

The background picture printing extension is compatible with all raster and PostScript printing devices. Its extension scope resource is defined as shown in Listing 2-21.

Listing 2-21 The extension scope resource for the background picture printing extension

```
resource gxExtensionScopeType (gxDriverScopeID,sysHeap,
                                                   purgeable)
{
   {
   'post',             /* compatible with PostScript printers */
   'rast'              /* compatible with raster printers */
   };
};
```

You can include multiple extension scope resources in your printing extension file to pinpoint the devices on which the extension runs. Typically, you use one extension scope resource to define the imaging systems that your extension is compatible with, and you add other extension scope resources to include or exclude specific devices. Listing 2-22 includes three extension scope resources.

Listing 2-22 An example of extension scope resources

```
resource gxExtensionScopeType (gxDriverScopeID, sysHeap,
                                                   purgeable)
{
   {
   'vect';        /* compatible with all vector devices */
   'post';        /* compatible with all PostScript devices */
   };
};

resource gxExtensionScopeType (gxPrinterScopeID, sysHeap,
                                                      purgeable)
{
   {
   'lwsc';        /* also compatible w/Personal LaserWriter SC */
   };
};

resource gxExtensionScopeType (gxPrinterExceptionScopeID,sysHeap,
                                                       purgeable)
{
   {
   'odd1';        /* not compatible with odd1 device */
   };
};
```

The resources in Listing 2-22 indicate that the printing extension is compatible with all PostScript and vector devices except for the one that is identified as `'odd1'` and with no raster devices except for the Personal LaserWriter SC.

Optimizing the Use of Your Extension

You must include an extension optimization (`'eopt'`) resource in your printing extension file to provide QuickDraw GX with information about when the extension needs to be loaded into memory. This information makes it possible to optimize the use of printing extensions, thus maximizing performance during the printing process. The optimization resource has an ID of `gxExtensionOptimizationID`.

QuickDraw GX also uses the values in this resource to determine if your extension has to be available on the user's system or on a print server if the user is printing from a print server on a network. For example, an extension that overrides messages during the despooling process needs to be available on the system that is communicating with the printer, while an extension that overrides messages during the spooling process needs to be available on the system that is creating the spool file. You need to carefully choose which of the optimization flags to include in the extension optimization resource for your printing extension. Including the wrong flag can cause performance degradation or worse, so choose carefully from among the flag values. The constants for these flags, each of which is a Boolean value, are shown in Table 6-11 on page 6-30 in the chapter "Printing Resources."

Listing 2-23 shows the extension optimization resource for the background picture printing extension, which modifies each page of a document as it is despooled.

Listing 2-23 An example of an extension optimization resource

```
resource 'eopt' (gxExtensionOptimizationID,sysHeap,purgeable)
{
   {
   gxDontExecuteDuringImaging,
   gxDontNeedDeviceStatus,
   gxChangePageAtGXDespoolPage,
   gxDontChangePageAtGXImagePage,
   gxDontChangePageAtGXRenderPage,
   gxNotServerPresenceRequired,
   gxClientPresenceRequired
   };
};
```

The despooling process during the imaging phase of printing is described in the chapter "Introduction to Printing Extensions and Drivers" in this book. The `GXDespoolPage` message is described on page 4-75, the `GXImagePage` message is described on page 4-94, and the `GXRenderPage` message is described on page 4-96 in the chapter "Printing Messages."

Specifying Which Messages Your Extension Overrides

You must include an override (`'over'`) resource in your printing extension file to provide QuickDraw GX with a list of the messages that your printing extension is overriding. Each entry in the override resource specifies a message and information about the resource in which the code segment for the message override is found. You typically include separate override resources for universal messages and for messages specific to an imaging system, as shown in Listing 2-24.

The code segment information for each message includes the resource ID of the code segment and the offset into the code segment for the instruction to jump to the message override function. The first 4 bytes of the code segment are reserved for use by QuickDraw GX and must be 0; thus, the first offset location is 4.

Each override resource in a printing extension has an ID of 0 for messages specific to an imaging system and an ID of `gxExtensionOverrideID` (-1) for universal messages. Unlike printer drivers, printing extensions may not override messages that QuickDraw GX sends to ensure compatibility with the Macintosh Printing Manager.

Listing 2-24 Override resources for a printing extension

```
#define segmentID NewSegmentID   /* defined as part of the make */
#define firstOffset 4
#define kRasterMsgsOVerrideID gxPrintingDriverBaseID+1

resource gxOverrideType (gxExtensionUniversalOverrideID, sysHeap,
                                                      purgeable)
{
   {
   gxInitialize,     segmentID, firstOffset+0,
   gxShutDown,       segmentID, firstOffset+4,
   gxDespoolPage,    segmentID, firstOffset+8,
   gxSetupImageData, segmentID, firstOffset+12,
   gxOpenConnection, segmentID, firstOffset+16,
   gxCloseConnection,segmentID, firstOffset+20,
   gxStartSendPage,  segmentID, firstOffset+24,
   gxFinishSendPage, segmentID, firstOffset+28,
   gxDefaultPrinter, segmentID, firstOffset+32,
   gxWriteData,      segmentID, firstOffset+36,
   gxCheckStatus,    segmentID, firstOffset+40,
   gxGetDeviceStatus,segmentID, firstOffset+44,
   gxCreateImageFile,segmentID, firstOffset+48,
   gxFetchTaggedData,segmentID, firstOffset+52,
   };
};

resource gxOverrideType (gxExtensionImagingOverrideID, sysHeap,
                                                   purgeable)
{
   {
   gxRasterDataIn,   segmentID, firstOffset+56,
   gxRasterLineFeed, segmentID, firstOffset+60,
   };
};
```

The first resource in Listing 2-24 lists the universal messages that the extension overrides, and the second resource specifies the messages that are specific to a raster imaging system and that the extension overrides. Each message listed in the resources specifies the name of the message, the ID of that segment, and where to find the message in the jump table. The ID of the code segment for all of these messages is defined by the constant `segmentID`. The jump statements to the code that implements the overrides are each 4 bytes long. The first is found at the first offset location (byte 4), and each subsequent jump statement is found 4 bytes beyond the previous one.

Defining the Loading Order of Your Extension

You also need to include an extension load (`'load'`) resource for your printing extension to tell QuickDraw GX the default value that specifies where to load your extension into the printing message chain. If you have one or more message overrides that prefer to handle a message before or after other handlers, you can specify that in this resource. The value that you define in your extension load resource specifies the default loading order for your extension when it is first added to the system. The user can modify the loading order of extensions by using the Extension Setup dialog box.

Note

The value that you specify in the extension load resource for your extension provides a default. It is used by QuickDraw GX as a hint. Because the user can change the order, you do not have any guarantee regarding the load order of your extension. u

The constants for the values that you can specify in your extension load resource are shown in Table 2-7.

Table 2-7 Loading order constants for an extension

Constant	Explanation
`gxExtensionLoadFirst`	This extension is loaded as the top message handler in the printing message chain
`gxExtensionLoadAnywhere`	This extension can be loaded anywhere in the print message chain because its message overrides don't depend on a handling order
`gxExtensionLoadLast`	This extension is loaded as the bottom message handler in the print message chain

The extension load resource for the background picture printing extension is shown in Listing 2-25.

Listing 2-25 The extension load resource for the background picture extension

```
resource gxExtensionLoadType (gxExtensionLoadID,sysHeap,purgeable)
{
   gxExtensionLoadFirst
};
```

The background picture extension draws the background picture on each page after the page has been fully composed. Thus, this printing extension should be loaded first in the message chain, which means that it receives messages after all other message handlers. Then the printer driver and any other message handler can modify the page contents at despool time prior to the background picture being drawn.

C H A P T E R 3

Printer Drivers

Contents

Printer Drivers

This chapter describes how you can develop printer drivers for printing documents with QuickDraw GX on printing devices. For example, you might develop a printer driver that allows a third-party printer or plotter to print documents created by any application that uses QuickDraw GX.

To use this chapter, you need to be familiar with how the printing features in QuickDraw GX work. *Inside Macintosh: QuickDraw GX Printing* describes these printing features and gives you an overview of the printing process.

Printer drivers include a number of resources. Before reading this chapter, you need some understanding of Macintosh resources and resource files. You can read about them in the Resource Manager chapter in *Inside Macintosh: More Macintosh Toolbox*.

Printer drivers use collections, which are managed by the Collection Manager. Drivers are activated by messages, which are managed by the Message Manager. You need to know about the Collection Manager and Message Manager components of QuickDraw GX to understand how to develop drivers. The chapter "Introduction to Printing Extensions and Drivers" in this book provides an overview of using these managers in printer drivers. Both of them are described in *Inside Macintosh: QuickDraw GX Environment and Utilities.*

This chapter begins with an overview of printer drivers and then discusses

- the tasks that any printer driver must accomplish
- the source files for a typical printer driver
- the printing messages used to create a printer driver
- the ImageWriter II printer driver as an example of how to construct a printer driver
- the resources used to create the ImageWriter II driver
- the user interface requirements for QuickDraw GX printer drivers

For specific reference information about the messages that you override to develop a printer driver, see the chapter "Printing Messages" in this book. For specific reference information about the resources that you need to include in your printer driver file, see the chapter "Printing Resources" in this book.

Note

The code samples presented in this chapter are taken from a recent version of the the sample code. These samples are not complete and may have been slightly modified since printing. You can find the latest version of the code in the QuickDraw GX sample code. u

About Printer Drivers

Macintosh system software uses device drivers to communicate information to hardware devices. A **printer driver** is a device driver—an independent software component that the system software uses to convert document data into printed output on a peripheral device such as a printer or plotter. In the QuickDraw GX environment, a printer driver sends QuickDraw GX data and instructions in a form specific to the printing device that it drives and manages the communications with that printing device.

Each printer driver has a Chooser interface through which the user can select communications parameters or special features of the driver. Each driver also displays status and alert information to the user through the Finder, as described in the section "Displaying Status Information and the Printing Alert Boxes" beginning on page 3-41.

QuickDraw GX allows the user to print documents to a different computer than is used to create the documents. This is often the case when a network includes a print server that several computers share. When the same computer is used, only one copy of your driver is required. When different computers are used, a copy of your printer driver must reside both on the computer that is sending the document to be printed and on the computer that is communicating with the printing device. The use of more than one copy of your driver is generally invisible to you, but understanding it does come into play because of how status and error conditions are handled in your driver. This is described in the section "Handling Status and Alert Conditions" beginning on page 3-9.

You need to develop a separate driver for each hardware device that has different characteristics. Developing a printer driver is different for each imaging system.

- **Raster imaging system** drivers convert QuickDraw GX shapes into raster data—blocks of data representing the pixels in an image—which are sent to raster printing devices to produce printed output. The Apple ImageWriter is an example of a raster printing device.
- **PostScript imaging system** drivers convert QuickDraw GX shapes into PostScript instructions, which are sent to PostScript printing devices to produce printed output. The Apple LaserWriter Pro 600 is an example of a PostScript printing device.
- **Vector imaging system** drivers convert QuickDraw GX shapes into vectors, which are sent to vector printing devices to produce printed output. Plotters that use HPGL are examples of vector printing devices.

A printer driver consists of resources that define the driver to the printing system. A printer driver is a message handler in a message chain. QuickDraw GX communicates with a driver by sending printing messages down the message chain. A driver receives a printing message by defining the set of messages it wishes to receive and act on. For example, a printer driver that supports raster printing devices overrides the messages for the raster imaging system.

Although you need to specify a fair amount of information in your printer driver resources, you can create drivers for many printing devices without writing very much code. QuickDraw GX provides default implementations of all of the tasks that a printer driver needs to accomplish, which means that you only need to override the printing messages that relate to specific or unique qualities of your printing device. For example, some PostScript printer drivers can use all of the default implementations of printing messages, with any device-specific parameters defined in the resource descriptions that you provide.

Each of the following sections describes a task that a QuickDraw GX printer driver needs to manage:

- writing document data to the spool-file when the application initiates the printing process
- reading document data from the spool file when creating the printed output
- converting QuickDraw GX shapes into a format that the printing device can understand
- handling the physical communication of data with the printing device
- interacting with the Finder to respond to Printing menu events
- providing a Chooser interface so that users can select your driver
- reporting status and alert conditions to the user
- providing compatibility with the Macintosh Printing Manager application interface

Each printing task is accomplished during one of the phases of printing, which are described in the chapter "Introduction to Printing Extensions and Drivers" in this book. Some of the printing tasks execute in the foreground and others execute in the background. A **foreground task** takes control of the computer, preempting the user from performing any other application tasks. A **background task** operates by taking control of the computer for brief periods of time while the application has relinquished time, allowing the user to continue working.

Writing Data to the Spool File

The first task that a QuickDraw GX printer driver needs to manage involves spooling a document to disk. During spooling, which occurs in the foreground, the application writes document data to a spool file by sending an intermediate representation of the document (which can be a portable digital document) to disk. Spooling occurs when the application calls the `GXStartJob` function, which causes QuickDraw GX to send the `GXStartJob` message.

Spooling of the document data to the spool file is handled by five messages: `GXCreateSpoolFile`, `GXSpoolPage`, `GXSpoolData`, `GXSpoolResource`, and `GXCompleteSpoolFile`.

The QuickDraw GX default implementations of the spooling messages provide the basic spooling functionality. You generally override these messages in a printing extension or printer driver to add specific data handling, such as encrypting the data that is sent to the spool file. If you are taking advantage of the default handling of these spooling messages, you always need to forward the messages before or after adding your own functionality.

If you choose to totally override a spooling message, that is, you choose not to forward it so that the default implementation can perform its tasks, you must handle all spooling on your own. This means that you must totally override all of the spooling messages and take care of writing the data to the spool file. You only need to do this if you are developing a driver that needs to create a custom spool-file format.

You can find a complete description of the interface and functionality of each of the spooling messages in the section “Spooling Messages” beginning on page 4-67 in the chapter “Printing Messages.”

Spooling Translated QuickDraw Data in Print Files

When spooling a file that contains QuickDraw drawing commands, QuickDraw GX by default stores untranslated QuickDraw data in the print file. When the print file is despooled and imaged, QuickDraw GX then converts the data to QuickDraw GX shapes and prints them. If you want to force QuickDraw GX to translate QuickDraw data to QuickDraw GX shapes before spooling, you can override the `GXFetchTaggedData` message that QuickDraw GX sends at spooling time to fetch resource data of type `'pcfg'`. Your override should forward the message, and then clear the highest-order bit of the data that is returned from the message. If the data’s highest-order bit is cleared, QuickDraw GX translates and stores the data in the print file as QuickDraw GX shapes. For more information on QuickDraw data in print files, see the discussion of the `'pict'` tag object in the advanced printing features chapter of *Inside Macintosh: QuickDraw GX Printing.* u

Reading Data From the Spool File

During the first part of the imaging phase of printing, QuickDraw GX despools each document from disk. During despooling, which occurs in the background, your printer driver reads the spooled document from the spool-file. Depooling of the document data from the spool-file is handled by five messages, which are complementary to the spooling messages: `GXCountPages`, `GXDespoolPage`, `GXDespoolData`, `GXDespoolResource`, and `GXCloseSpoolFile`.

The QuickDraw GX default implementations of the despooling messages provide the basic despooling functionality. You generally override these messages in a printing extension or printer driver to add specific data handling, such as decrypting the data that you encrypted in your overrides of the spooling messages. If you are taking advantage of the default handling of these despooling messages, you always need to forward the messages before or after adding your own functionality.

If you choose to totally override a despooling message, that is, you choose not to forward it so that the default implementation can perform its tasks, you must handle all despooling on your own. This means that you must totally override all of the despooling messages and take care of reading the data from the spool-file. As with spooling, you only need to do this if you are implementing a driver that needs to create a custom spool-file format.

You can find a complete description of the interface and functionality of each of the despooling messages in the section “Despooling Messages” beginning on page 4-74 in the chapter “Printing Messages.”

Despooling Print Files Containing QuickDraw Picture Data

When a document containing QuickDraw imaging commands is spooled, QuickDraw GX by default saves the QuickDraw data for each page in a tag object of tag type `'pict'` attached to a rectangle shape. If you examine the contents of a despooled file, note that the presence of a rectangle shape with such an attached tag object indicates the presence of QuickDraw picture data. For more information, see the discussion of the `'pict'` tag object in the advanced printing features chapter of *Inside Macintosh: QuickDraw GX Printing.* u

Converting QuickDraw GX Shapes

During the second part of the imaging phase of printing, your printer driver needs to convert the QuickDraw GX shape or shapes that compose each page into instructions that your printing device can use to produce the printed version of the page. This process of rendering each page into printable form takes place in the background. Different kinds of printing messages are sent during rendering:

- Universal imaging messages are always sent during the rendering phase. The universal imaging messages include `GXSetupImageData`, `GXImageJob`, `GXImageDocument`, `GXImagePage`, and `GXRenderPage`.
- Raster imaging messages are only sent when a user is printing to a device that uses the raster imaging system. The raster imaging messages are: `GXRasterDataIn`, `GXRasterLineFeed`, and `GXRasterPackageBitmap`.
- PostScript imaging messages are only sent when a user is printing to a device that uses the PostScript imaging system. Some of the PostScript imaging messages are: `GXPostScriptQueryPrinter`, `GXPostScriptGetPrinterGlyphsInformation`, `GXPostScriptEjectPage`, and `GXPostScriptProcessShape`.
- Vector imaging messages are only sent when a user is printing to a device that uses the vector imaging system. The vector imaging messages are: `GXVectorPackageShape`, `GXVectorLoadPens`, and `GXVectorVectorizeShape`.

The QuickDraw GX default implementations of the imaging messages provide the basic imaging functionality. You generally override some of these messages to manage the specifics of your device. Although you can override any of the messages to customize your driver, you often only need to override the `GXSetupImageData` universal imaging message to initialize your device, and then override some of the messages specific to the imaging system to handle your device.

If you are developing a printer driver for a PostScript device, you can likely use the default implementations for many PostScript devices, whereas you usually have to override the raster and vector imaging messages to generate the unique data sequences required by each raster and vector device.

You can find a complete description of the interface and functionality of each of the imaging messages in the chapter “Printing Messages” in this book.

Communicating With the Printing Device

During the device communications phase of printing, QuickDraw GX sends the data that composes the rendered form of each page to the printing device. This phase takes place in the background and is the phase during which physical communications with your device takes place.

Note
The device communications phase of printing can take place on a different computer than the computer on which imaging takes place. You need to keep this in mind when writing your code. For example, the code that you use to access a file during this phase needs to take this fact into account. u

QuickDraw GX normally handles device communications with these operations:

- Initiates communications with the `GXOpenConnection` message and terminates communications with the `GXCloseConnection` message.
- Initiates the sending of data for a page with the `GXStartSendPage` message and terminates sending of the page data with the `GXFinishSendPage` message.
- Adds data for a page to an output buffer with the `GXBufferData` message. QuickDraw GX sends the buffered data to the printing device with the `GXDumpBuffer` message. QuickDraw GX then sends the `GXFreeBuffer` message to wait for the completion of buffer processing.
- Sends data directly to the printing device (without buffering) with the `GXWriteData` message.

The QuickDraw GX default implementations of the device communications messages handle communicating with your printing device for you. QuickDraw GX provides different versions of these messages for the different kinds of standard I/O: serial, Printer Access Protocol (PAP), or not-connected. You specify the communications type of your printing device in your driver resources, as described in the chapter “Printing Resources” in this book.

You override the printing device communications messages when you need to customize the handling of the communication. For example, you can override the `GXOpenConnection` message to verify that a printing device of your type is connected to the computer and that the printing device is working properly. If you are writing a driver to capture printer output and store it in a file rather than directly communicate it to a printing device, you can override the `GXBufferData` message and store the buffered data in a file. If you need to modify the primitive I/O handling for your printing device, you can override the `GXWriteData` message.

You usually forward these messages after adding your own instructions about handling them. If you choose to create your own communications buffering scheme, you can totally override the `GXBufferData`, `GXDumpBuffer`, and `GXFreeBuffer` messages.

You can find a complete description of the interface and functionality of each of the printing device communications messages in the section “Device Communications Messages” beginning on page 4-131 in the chapter “Printing Messages.”

Interfacing With the Finder

The Macintosh Finder sends five messages that you might need to override in your driver: `GXGetDTPMenuList`, `GXDTPMenuSelect`, `GXInitializeStatusAlert`, `GXHandleAlertEvent`, and `GXHandleAlertFilter`. You override these messages to add any items to the Printing menu of the Finder and to handle events that occur in those items.

You can find a complete description of the interface and functionality of each of these Printing menu messages in the section "Finder Menu Messages" beginning on page 4-169 in the chapter "Printing Messages." Desktop printers and the menu choices associated with them are described in *Inside Macintosh: QuickDraw GX Printing*.

Providing a Chooser Interface

You need to make it possible for the user to choose your driver in the Chooser and to set any options that you provide, including various possibilities for communicating with your printing device, such as port options and baud rates.

You specify the connection possibilities for your driver in a look (`'look'`) resource, which is used by the Chooser to display choices to the user. The look resource references the communications (`'comm'`) resources that you provide, which allows QuickDraw GX to configure the device communications after the user makes a selection. The look and communications resources are discussed in the section "Using Resources in Drivers" beginning on page 3-53 and are described in detail in the chapter "Printing Resources" in this book.

You also need to define Chooser package (`'PACK'`) and list definition (`'LDEF'`) resources for the Finder interface to your printer driver, just as you do for any other Macintosh driver. These resources are described in the Finder interface chapter in *Inside Macintosh: More Macintosh Toolbox*.

Handling Status and Alert Conditions

During the device communications phase of printing, you need to provide status information to the user through the Finder, and you need to handle error or alert conditions that occur on your printing device.

Status information is displayed in the desktop printer window, which is provided by the Finder. While your printer driver is handling the printing of a job, you update the status information that is displayed in this window with text strings such as "Downloading fonts to printer" or "Opening connection."

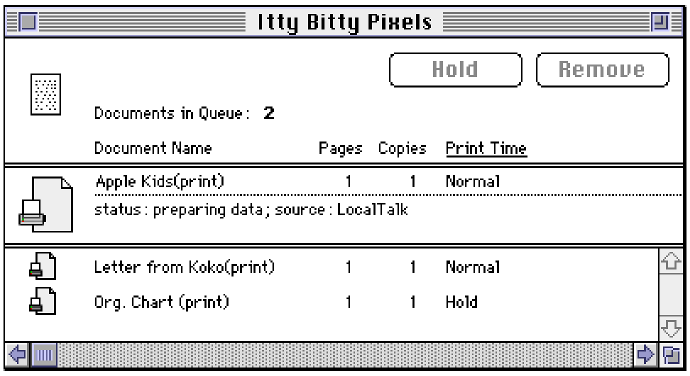
When a user is printing on a network with a print server, two copies of your driver are in use at the same time: one on the computer on which the user initiated printing (the client), and another on the server that is connected to the printing device. Several user machines can be communicating with the print server simultaneously, so the Finder arbitrates which status information is sent to which driver on which user machine.

In status (`'stat'`) resources that you include with your driver, you define the status text. When you want to display status information to the user, you call the

GXReportStatus function, including the ID of the status text that you want shown. The Finder receives the status information from the GXReportStatus function and sends a GXWriteStatusToDTPWindow message to the copy of your driver that is running on the machine that initiated printing. The default implementation of the GXWriteStatusToDTPWindow message converts the status text ID into a string (by accessing the resource) and calls the GXWriteDTPData function to display the string.

Figure 3-1 shows the display of a status text string in the desktop printer window.

Figure 3-1 The printing status displayed in a desktop printer window



Documents in Queue: 2

Document Name	Pages	Copies	Print Time
Apple Kids(print)	1	1	Normal
status: preparing data; source: LocalTalk			
Letter from Koko(print)	1	1	Normal
Org. Chart (print)	1	1	Hold

You can override the GXWriteStatusToDTPWindow message if you need to add information to the status text string at run time. For example, if you are composing a status text string that includes the current page number, you could override this message. In your override, you would determine if the current status text string is the one that you want to change, and produce a text string that incorporates the page number into it.

You need to alert the user when certain conditions arise during the printing of a document; for example, when the printing device runs out of paper, when the printing device requires that a sheet of paper be fed manually, or when a hardware error has occurred.

When these kinds of conditions occur, you have to alert the user, who then must take an action before printing can continue. The standard way to handle this is with a printing alert box. QuickDraw GX can manage displaying your printing alert box when you define printing alert ('plrt') resources. These resources automate the creation and display of alert boxes.

You implement handling of a printing alert box with the GXAlertTheUser function, as described in the chapter “Printing Functions for Message Overrides” in this book. An example of using the GXAlertTheUser function is shown in Listing 3-15 on page 3-42.

You can find a complete description of the interface and functionality of each of the status and error-handling messages in the chapter “Printing Messages” in this book. You can find descriptions of the status and printing alert resources in the chapter “Printing Resources” in this book.

Providing Compatibility With the Macintosh Printing Manager

A final task to manage in your printer driver is how to interface to applications that use the Macintosh Printing Manager in the Macintosh Operating System. QuickDraw GX provides a default translation from Macintosh Printing Manager calls into QuickDraw GX messages, which allows those applications to use your printer driver without change.

You can customize this conversion with several resources. You can also override the default implementations of the Macintosh Printing Manager compatibility messages, which include such messages as `GXPrOpenDoc`, `GXPrCloseDoc`, `GXPrOpenPage`, `GXPrJobInit`, `GXPrValidate`, and `GXPrJobMerge`.

You can find a complete description of the interface and functionality of each of the Printing Manager compatibility messages in the section “Compatibility Messages” beginning on page 4-147 in the chapter “Printing Messages.” You can find a description of the compatibility resources in the section “Resources Used for Printing Extensions and Printer Drivers” beginning on page 6-12 in the chapter “Printing Resources.”

Writing Printer Driver Files

After you know which messages you need to override as part of managing certain printing tasks, you need to write your printer driver in three kinds of files:

- You need to write a jump table in an assembly-language file so that QuickDraw GX can invoke the correct code in response to the printing messages that it sends.
- You need to write your message override functions in code files (typically, C language files).
- You need to define in Macintosh resource files the resources that contain the data for your printer driver.

The content of each of these file types is reviewed in this section, with sample code from the ImageWriter II printer driver included. The complete contents of the ImageWriter II driver files are found in the QuickDraw GX sample code.

Each driver must include one or more assembly-language jump tables, one or more files of code (in assembly-language, C, Pascal, or some other language) to write the functions that the driver performs, and one or more resource files. The ImageWriter II printer driver consists of the files shown in Table 3-1.

Table 3-1 Files used to implement the ImageWriter II printer driver

Filename	Contents
`CommonDefines.h`	The header file containing values and types used in other files
`ChooserSupport.a`	The assembly-language defines for working with the Chooser-related resources
`ChooserSupport.c`	The code for the resources used by the Chooser
`ChooserSupport.r`	The resources needed for the driver to work with the Chooser
`newapp.a`	The jump table for the overrides provided in the `newapp.c` module
`newapp.c`	The implementations of the QuickDraw GX printing messages that this driver overrides
`newapp.r`	The resources needed for the QuickDraw GX printing messages
`oldapp.a`	The jump table for the overrides provideded in the `oldapp.c` module
`oldapp.c`	The implementations of message overrides for compatibility with the Macintosh Printing Manager
`oldapp.r`	The resources needed for the Macintosh Printing Manager compatibility messages

This section describes the contents of several of these files. The complete files are found in the QuickDraw GX sample code.

The ImageWriter II printer driver is divided into two portions:

- The portion that provides the overrides of the QuickDraw GX printing messages is found in the files `newapp.a`, `newapp.c`, and `newapp.r`.
- The portion that provides the overrides of the Macintosh Printing Manager compatibility messages is found in the files `oldapp.a`, `oldapp.c`, and `oldapp.r`.

You can develop your printer driver in any number of files. Some developers find it convenient to separate the Macintosh Printing Manager compatibility portion of the code from the QuickDraw GX printing message portion, while others use different criteria to decide which code goes in which file.

Message Overrides and the Jump Table

You need to tell QuickDraw GX where (in which segment and at what offset from the beginning of that segment) to find the code for each printing message that you are overriding. You do this by defining an assembly-language jump table. The contents of the file `newapp.a`, which defines the jump table for the ImageWriter II printer driver, are shown in the QuickDraw GX sample code.

The jump table links the appropriate function into your code and provides a jump statement to invoke that function. The override (`'over'`) resource tells QuickDraw GX where to look (at which offset) in the jump table to find the jump statement for a specific printing message name. In this way, QuickDraw GX knows which of your functions to call to respond to the messages in which you are interested.

You define a jump table with one jump (`JMP`) statement for each message that you override. You also define an override resource that specifies the offset in that jump table for each message. The jump table and override resource must be coordinated. QuickDraw GX dispatches a printing message to your printer driver by jumping to the routine whose location is specified in the appropriate location in the jump table. The override resource is described in the section “The Override ('over') Resource” beginning on page 6-13 in the chapter “Printing Resources.”

QuickDraw GX reserves the first 4 bytes for its own use, so the value of the constant `firstOffset` is 4. These first 4 bytes are used by QuickDraw GX to maintain an owner count and must be set to 0 in the jump table.

For example, the ImageWriter II printer driver overrides the messages that are shown in Table 3-2. This driver also overrides several Macintosh Printing Manager compatibility messages, which are not shown in this table.

Table 3-2 Printing messages overridden by the ImageWriter II printer driver

Message	Why you override
`GXInitialize`	To set up the environment so that the printer driver can use predefined global values
`GXShutDown`	To free the storage that was allocated by the `GXInitialize` function
`GXDefaultPrinter`	To modify the default printer object characteristics, such as adding view devices that the application can use
`GXDefaultJob`	To modify the default job characteristics, such as which format is the default format
`GXJobDefaultFormatDialog`	To modify the behavior or appearance of the Page Setup dialog box
`GXJobFormatModeQuery`	To provide support for direct-mode printing
`GXOpenConnection`	To establish connection with the printing device
`GXSetupImageData`	To modify the initialization data that is used for imaging the entire job

continued

Table 3-2 Printing messages overridden by the ImageWriter II printer driver (continued)

Message	**Why you override**
`GXStartSendPage`	To perform any actions required to set up printing for the next page, such as alerting the user to manually feed a piece of paper
`GXRenderPage`	To perform tasks during the imaging of each page
`GXRasterPackageBitmap`	To perform your own translation from bitmaps into character sequences for your device
`GXRasterLineFeed`	To send the codes to the printing device that cause it to move the print head

The override resources for the ImageWriter II printer driver must include an entry for each of these printing messages, and the jump table must include a jump statement for each. Listing 3-1 shows the override resource definitions from the `newapp.r` file. The override resource definition for the Macintosh Printing Manager compatibility message overrides, from the `oldapp.r` file, is shown in the QuickDraw GX sample code.

Listing 3-1 Two override resources from the ImageWriter II printer driver

```
#define firstOffset 4
#define segmentID NewSegID

resource gxOverrideType (gxDriverUniversalOverrideID, sysHeap,
                                                      purgeable)
{
   {
   gxInitialize,              segmentID, firstOffset + 0,
   gxShutDown,                segmentID, firstOffset + 4,
   gxDefaultPrinter,          segmentID, firstOffset + 8,
   gxDefaultFormat,           segmentID, firstOffset + 12,
   gxDefaultJob,              segmentID, firstOffset + 16,
   gxJobDefaultFormatDialog,  segmentID, firstOffset + 20,
   gxJobFormatModeQuery,      segmentID, firstOffset + 24
   gxRenderPage,              segmentID, firstOffset + 28,
   gxOpenConnection,          segmentID, firstOffset + 32,
   gxCloseConnection,         segmentID, firstOffset + 36
   gxStartSendPage,           segmentID, firstOffset + 40,
   gxSetupImageData,          segmentID, firstOffset + 44,
   };
};

/*
```

```
   The following are overrides for raster-specific messages,
   including where to find them in the jump table.
*/
resource gxOverrideType (gxDriverImagingOverrideID, sysHeap)
{
   {
   gxRasterPackageBitmap,  segmentID, firstOffset + 48,
   gxRasterLineFeed,       segmentID, firstOffset + 52,
   };
};
```

In Listing 3-1, the name of each message is followed by the ID of the segment in which its code resides (in this case, the constant `segmentID`) and the byte offset of its jump statement in the jump table. QuickDraw GX reserves the first 4 bytes for its own use, so the value of the constant `firstOffset` is 4. These first 4 bytes are used by QuickDraw GX to maintain an owner count and must be set to 0 in the jump table.

The ImageWriter II printer driver includes two override resources for these printing messages, as shown in Listing 3-1: the first defines which universal printing messages the driver overrides, and the second defines which printing messages the driver overrides that are specific to the raster imaging system. You need to define the universal message overrides in a resource with a different ID than the resource in which you define the message overrides that are specific to the imaging system. This is described in the chapter "Printing Resources" in this book. The ImageWriter II driver also includes a third override resource for the Printing Manager compatibility messages that it overrides.

You implement the jump table as an assembly-language program that contains a jump statement for each override function. You need to list the jump statements in exactly the same order as you listed the message names in your override resource; otherwise, QuickDraw GX will invoke the wrong function in response to a message. Listing 3-2 shows the jump table for the QuickDraw GX messages found in the override resources in the `newapp.r` file.

Listing 3-2 The jump table for the ImageWriter II printer driver

```
SD_JumpTable   PROC  EXPORT

         DC.L  0

         ; Universal messages
         IMPORT SD_Initialize
         JMP    SD_Initialize

         IMPORT SD_ShutDown
         JMP    SD_ShutDown
```

```
        IMPORT SD_DefaultPrinter
        JMP    SD_DefaultPrinter

        IMPORT SD_DefaultFormat
        JMP    SD_DefaultFormat

        IMPORT SD_DefaultJob
        JMP    SD_DefaultJob

        IMPORT SD_JobDefaultFormatDialog
        JMP    SD_JobDefaultFormatDialog

        IMPORT SD_JobFormatModeQuery
        JMP    SD_JobFormatModeQuery

        IMPORT SD_RenderPage
        JMP    SD_RenderPage

        IMPORT SD_OpenConnection
        JMP    SD_OpenConnection

        IMPORT SD_CloseConnection
        JMP    SD_CloseConnection

        IMPORT SD_StartSendPage
        JMP    SD_StartSendPage

        IMPORT SD_SetupImageData
        JMP    SD_SetupImageData

        ; Raster messages
        IMPORT SD_PackageBitmap
        JMP    SD_PackageBitmap

        IMPORT SD_LineFeed
        JMP    SD_LineFeed

        END
```

Note

The code shown in Listing 3-2 is for the MPW environment. If you are programming in a different development environment, you might need to use different assembler directives. You must, however, be certain to include the initial 4 bytes (set to 0) and each JMP statement. u

The EXPORT statement at the beginning makes the jump table public. The IMPORT statements make it possible for your assembly-language program to reference the C language functions that are performing your message overrides and to provide correct linkage to those functions.

Because QuickDraw GX uses the first 4 bytes in the jump table to store the owner count value, the first statement (DC.L) must be included. These bytes must all have the value 0 in them.

The name of each override function provided by the ImageWriter II printer driver is prefixed with the string SD_ to differentiate it from other overrides of the same message. This means that the driver's override of the GXInitialize message is named SD_Initialize, its override of the GXOpenConnection message is named SD_OpenConnection, and so on.

You can choose to intersperse the IMPORT and JMP statements as shown in Listing 3-2, or you can place all of the IMPORT statements together, followed by all of the JMP statements, as is shown in Listing 2-2 on page 2-10 in the chapter "Printing Extensions."

IMPORT ANT

Always coordinate the entries in your override resources with the entries in your jump table. If they are not aligned, the wrong code will be executed for a message. The offset that you specify in the resource for each message must match the offset of the corresponding function in your jump table. You must also include 4 bytes with zero values at the beginning of your jump table. s

The Message Overrides

The code that makes up your printer driver is a set of functions that override, either partially or totally, some of the printing messages that QuickDraw GX sends during the process of printing a document. This section describes how you use printing messages to develop your driver. It also describes how to use the printing collections to access and modify printing information, and how to use the nrequire macro to handle exceptions in your code.

Whenever you override a QuickDraw GX printing message, you must be certain that the declaration of your override function matches the declaration of the message. This means that the type of function return and the type of each parameter must match the types in the message declaration. The chapter "Printing Messages" shows the declaration of each printing messages.

QuickDraw GX provides a default implementation of each printing message that it sends. You can augment (partially override) some messages, and you can replace (totally override) others. For each printing message, QuickDraw GX provides one of three kinds of default implementations, as shown in Table 2-3 on page 2-12 in the chapter "Printing Extensions."

The contents of the file newapp.c, which contains the source code for the QuickDraw GX message overrides in the ImageWriter II printer driver, are shown in the QuickDraw GX sample code.

Choosing the Messages to Override

Which printing messages you need to override in your printer driver depends entirely on the characteristics of your device. Although QuickDraw GX provides default implementations of most messages (which means that you are not required to override them), there are some messages that you must override for raster or vector printing devices. Some drivers override many messages to provide their operations, and other drivers are created by overriding only a few messages.

Note

The action of the default implementation of each printing message is noted in the chapter "Printing Messages." Some of the default implementations provided by QuickDraw GX are empty (all that the function does is return). u

Your message overrides for a different printing device are likely to be quite different than those for the ImageWriter II, which are shown in Table 3-2 on page 3-13. For example, the driver for the LaserWriter IIg creates a PostScript file during the printing process. This driver only needs to override a few messages to perform its tasks, as shown in Table 3-3.

Table 3-3 Message overrides for the LaserWriter IIg printer driver

Message	Why you override
`GXPostScriptDoPageSetup`	To add a PostScript setup string at the start of data for each page
`GXOpenConnection`	To create a file for the PostScript data
`GXCloseConnection`	To close the PostScript data file
`GXInitialize`	To allocate the message globals
`GXDumpBuffer`	To write the printer data to the file

The messages that your printer driver needs to override depend on the unique characteristics of your printing device. There is no set formula: what you have to do is familiarize yourself with the messages that are available from the printing system and look at the code for sample printer drivers. You can review the code for the sample drivers in the QuickDraw GX sample code, and you can examine the printing messages in the chapter "Printing Messages" in this book.

Forwarding Messages

Your printer driver can forward a message in its implementation of a partial override. If you are totally overriding a message, you do not forward it to other message handlers. You can forward the message either before or after performing your own actions. To forward a message to the next message handler in the message chain, use a statement with the following format:

```
anErr = Forward_MessageName(arguments);
```

For example, the ImageWriter II printer driver overrides the `GXRasterLineFeed` message to test whether it is printing in low resolution. If so, the override function, `SD_LineFeed`, temporarily alters the value of one of the parameters, forwards the message, and then resets the parameter value. Listing 3-3 shows the override of the `GXRasterLineFeed` message from the ImageWriter II printer driver.

Listing 3-3 Overriding the `GXRasterLineFeed` message

```
OSErr SD_LineFeed (Int16 *lineFeedSize, Ptr buffer,
                   UInt32 *bufferPos,
                   gxRasterImageDataHdl hImageData )
{
   OSErr    anErr;
   Boolean  amLowRes;
   short    actualLineFeed = *lineFeedSize;

   amLowRes = ((**hImageData).vImageRes == ff(72));
/*
   If the user is printing in low-resolution mode, double the
   line-feed size because the ImageWriter line-feed commands are
   all expressed at 144 dpi.
*/
   if (amLowRes)
         *lineFeedSize <<= 1;
/*
   To get rid of the “paper dance” for blank color passes,
   optimize the small motions into groups.
*/
   {
   SpecGlobalsHdl    hGlobals= GetMessageHandlerInstanceContext();
   SpecGlobalsPtr    pGlobals= *hGlobals;

   if ( (pGlobals->packagingOptions == kDoSmallLineFeeds) ||
         (*lineFeedSize < -1) ||
         (*lineFeedSize >  1) )
      {
      *lineFeedSize += pGlobals->lineFeeds;
      pGlobals->lineFeeds = 0;
      /* do the line feed in the default way */
      anErr = Forward_GXRasterLineFeed(lineFeedSize, buffer,
                              bufferPos, hImageData);
      }
   else
      {
      pGlobals->lineFeeds += *lineFeedSize;
      *lineFeedSize = 0;
      anErr = noErr;
      }
   }

   if (amLowRes)
      *lineFeedSize >>= 1;
   return(anErr);
}
```

The SD_LineFeed override function tests the printing resolution. If the user is printing at low resolution (72 dots per inch), then the line-feed value is temporarily doubled to accommodate the fact that all line feeds are expressed in units of 144 dots per inch on the ImageWriter II. This function then forwards the GXRasterLineFeed message to allow the default implementation to send the appropriate character sequences. The GXRasterLineFeed message is described on page 4-98 in the chapter "Printing Messages."

Sending Messages

Your printer driver can also send a printing message to other message handlers. When you send a message, QuickDraw GX receives it and then sends it to the first message handler in the chain. To send a message, use a statement with this format:

```
anErr = Send_GXMessageName(arguments);
```

For example, to send the GXFreeBuffer message, use the following statement:

```
anErr = Send_GXFreeBuffer(bufferPtr);
```

Handling Exceptions in Your Message Overrides

The code samples presented in this chapter make use of an exception-handling strategy that simplifies the job of testing for error conditions after each function call. This strategy uses three C macros to branch to error handlers in response to conditions. The error-handling macros are described in the section "Handling Exceptions in Your Message Overrides" beginning on page 2-14 in the chapter "Printing Extensions."

Using the Printing-Related Collections

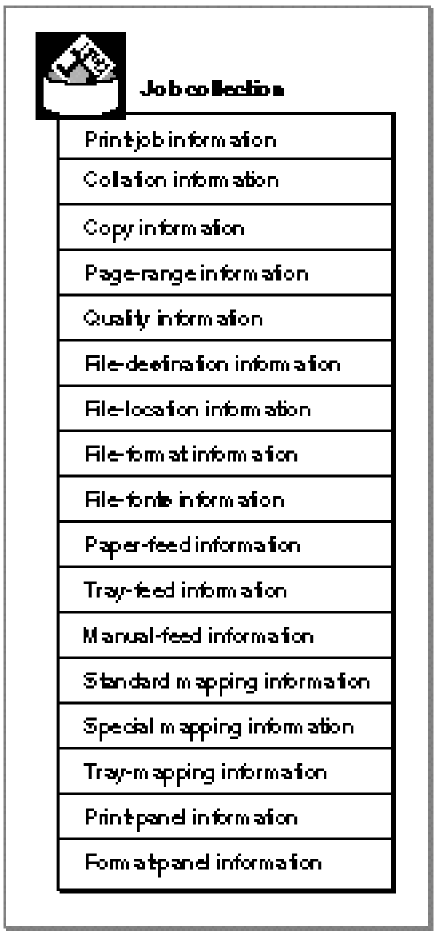
Much of the data that defines how a document is processed for printing is stored in collection objects. QuickDraw GX supports three collections to store printing-related information:

- The job collection contains information from the Print dialog box.
- The format collection contains information from the Page Setup and Custom Page Setup dialog boxes.
- The paper-type collection stores information specified by the creator of the paper-type object.

These collections are described in *Inside Macintosh: QuickDraw GX Printing*. Collections and collection items are described in the chapter "Collection Manager" in *Inside Macintosh: QuickDraw GX Environment and Utilities.*

Most of the collection items with which you need to work with to develop a printer driver are found in the job collection, which is illustrated in Figure 3-2. You access the information in this collection when your driver needs to modify the choices that are presented to the user and when your driver needs to access the current settings of various printing parameters.

Figure 3-2 The job collection



Each of the job collection items shown in Figure 3-2 is described in *Inside Macintosh: QuickDraw GX Printing.*

You can access each item in this collection by specifying its tag ID. Using tags to access collection items is described in *Inside Macintosh: QuickDraw GX Environment and Utilities.* The tag ID that you use for each item is listed in Table 3-4.

Table 3-4 Tag ID constants for items in the job collection

Constant	Value	Explanation
`gxJobTag`	`'job'`	Print-job information
`gxCollationTag`	`'sort'`	Collation information
`gxCopiesTag`	`'copy'`	Copy information
`gxPageRangeTag`	`'rang'`	Page-range information
`gxQualityTag`	`'qual'`	Quality information
`gxFileDestinationTag`	`'dest'`	File-destination information
`gxFileLocationTag`	`'floc'`	File-location information
`gxFileFormatTag`	`'ffmt'`	File-format information
`gxFileFontsTag`	`'incf'`	File-fonts information
`gxPaperFeedTag`	`'feed'`	Paper-feed information
`gxTrayFeedTag`	`'tray'`	Tray-feed information
`gxManualFeedTag`	`'manf'`	Manual-feed information
`gxNormalMappingTag`	`'nmap'`	Standard mapping information
`gxSpecialMappingTag`	`'smap'`	Special mapping information
`gxTrayMappingTag`	`'tmap'`	Tray-mapping information
`gxPrintPanelTag`	`'ppan'`	Print-panel information
`gxFormatPanelTag`	`'fpan'`	Format-panel information

Each of the items in the job collection has a specific data structure associated with it, and most have a number of constants defined for setting the values of fields in the structures. These data structures and constants are described in the chapter “Page Formatting and Dialog Box Customization” in *Inside Macintosh: QuickDraw GX Printing.*

s **WARNING**
The size of the items in the job collection is subject to change as QuickDraw GX evolves. For that reason, it is important for you to specify an expected size for each item in your calls to the Collection Manager, rather than specifying `nil`, which tells the Collection Manager to copy the entire object no matter what its size is. The size that you specify must match the size of the data structure into which the item is being copied. s

The ImageWriter II printer driver accesses several of the job collection items to determine how to proceed with printing. One example, the `JobIsBest` function, is shown in Listing 3-12 on page 3-38. This function accesses the quality information to determine how to send data to the printing device.

The QuickDraw GX ImageWriter II Printer Driver Messages

This section describes the messages and functions that make up the QuickDraw GX ImageWriter II printer driver. This driver needs to accomplish the following tasks, several of which are specific to raster printing:

- Initialize the environment.
- Modify the default printer object so that the application can choose to format for black-and-white or color printing at different resolution settings.
- Establish highest-resolution text printing as the preferred job format mode for the ImageWriter II.
- Update the printing-device configuration information and store it with the desktop printer.
- Respond to queries about direct-mode printing.
- Set up the data that is needed for printing the document. For the ImagerWriter II, this means setting up information, including
  - the draft character table if the document is printing in text-draft mode
  - the halftone data for printing graphics
  - unidirectional or bidirectional printing, depending on the selected resolution
  - the raster packaging data, depending on the printing resolution
  - print-quality information
  - color-printing information
- Render each page by creating a buffer for the data and then sending it to the printing device.
- For each page in the document being printed, check to see if the page needs to be manually fed by the user. If so, alert the user to feed the appropriate paper type, and manage the user’s interaction with the printing alert box.
- Shut down the environment at the end of printing.

The messages that the ImageWriter II printer driver overrides are shown in Table 3-2 on page 3-13. The remainder of this section describes these messages and, as an example, shows how the ImageWriter II driver overrides them.

Initializing the ImageWriter II Environment

QuickDraw GX sends the `GXInitialize` message when an application creates a job object, just after the application has called the `GXNewJob` function so that your driver can initialize any storage and establish the values of global variables. The ImageWriter II printer driver performs very simple initialization. Its override of the `GXInitialize` message, `SD_Initialize`, allocates a globals world so that the driver can access the QuickDraw GX globals. The `SD_Initialize` function is shown in Listing 3-4.

Listing 3-4 Initializing the ImageWriter II printer driver

```
OSErr SD_Initialize (void)
{

   SpecGlobalsHdl hGlobals;
   OSErr          anErr;

   /* create the globals */
   hGlobals = (SpecGlobalsHdl) NewHandleClear(
                                      sizeof(SpecGlobals) );
   anErr = MemError();

   /* save the globals */
   SetMessageHandlerInstance(hGlobals);

   /* branch to exception handler if necessary */
   nrequire( anErr, MNewHandleClear );

   /* no draft table allocated yet */
   (**hGlobals).draftTable = nil;

   return(noErr);

MNewHandleClear:
   return(anErr);
}
```

The `SD_Initialize` function first allocates the QuickDraw GX message globals. Then it initializes the global handle to the draft table to `nil` because this table, which is used for drawing draft-quality characters on the printing device, is only needed when the ImageWriter II is printing in draft mode.

You almost always override the `GXInitialize` message for a driver. No matter which imaging system your driver is written for, you need to initialize the QuickDraw GX message globals. Th `GXInitialize` message is described on page 4-43 in the chapter "Printing Messages."

Providing the Application With Printing Options

QuickDraw GX sends the `GXDefaultPrinter` message when an application creates a job object so that your driver can create new view devices that represent the specific features provided by your printing device. These features include color information. For raster printing devices, the features also include resolution variations; for vector printing devices, the features include pen information. You can override this message to create view devices for your driver. You attach these view devices to the job object, and the application can then access them to determine specific device settings.

The ImageWriter II printer driver provides various printing resolutions and allows the application to print either in black and white or in color. The driver overrides the `GXDefaultPrinter` message to add two high resolution (144 dpi) view devices for use by applications. This message override, the `SD_DefaultPrinter` function, is shown in Listing 3-5. The `GXDefaultPrinter` message is described on page 4-50 in the chapter "Printing Messages."

Listing 3-5 Modifying the default printer object

```
OSErr SD_DefaultPrinter(gxPrinter thePrinter)
{
   OSErr          anErr;
   gxViewDevice   vd;

   /* add the standard view devices first */
   anErr = Forward_GXDefaultPrinter(thePrinter);
   nrequire(anErr, DefaultPrinter);

   /* add a 144 b/w view device */
   vd = NewDeviceResolutionViewDevice();

   {
   gxSetColor  theColors[2];
   gxSetColor  *pColor;
   gxColorSet  theSet;
   pColor = &theColors[0];

   pColor->rgb.red   = 0xFFFF;
   pColor->rgb.green = 0xFFFF;
   pColor->rgb.blue  = 0xFFFF;

   pColor++;
   pColor->rgb.red   = 0x0000;
   pColor->rgb.green = 0x0000;
   pColor->rgb.blue  = 0x0000;

   theSet = GXNewColorSet(gxRgbSpace, 2, theColors);
   SetViewDeviceColorSet(vd, theSet);
   GXDisposeColorSet(theSet);
   }

   anErr = GXAddPrinterViewDevice(thePrinter, vd);
   nrequire(anErr, FailedAddBWViewDevice);

   /* add a 144 color view device with 8 colors */
   if (PrinterHasColorRibbon())
      {
      gxSetColor  theColors[8];
      gxSetColor  *pColor;
      gxColorSet  theSet;
      short       idx;

      vd = NewDeviceResolutionViewDevice();
      pColor = &theColors[0];
      for (idx = 0; idx < 8; ++idx)
         {
         /* default the color to black */
         pColor->rgb.red   = 0x0000;
         pColor->rgb.green = 0x0000;
         pColor->rgb.blue  = 0x0000;
         /* then fill in the RGB components */
         if (idx & 0x04)
            pColor->rgb.red   = 0xFFFF;
         if (idx & 0x02)
            pColor->rgb.green = 0xFFFF;
         if (idx & 0x01)
            pColor->rgb.blue  = 0xFFFF;
         ++pColor;
```

```
            /* move onto the next color */
            }
        theSet = GXNewColorSet(gxRgbSpace, 8, theColors);
        SetViewDeviceColorSet(vd, theSet);
        GXDisposeColorSet(theSet);

        anErr = GXAddPrinterViewDevice(thePrinter, vd);
        nrequire(anErr, FailedAddColorViewDevice);
        }

    return(noErr);

/* exception handling */
FailedAddColorViewDevice:
FailedAddBWViewDevice:
    GXDisposeViewDevice(vd);

GXDefaultPrinter:
    return(anErr);
}
```

The `SD_DefaultPrinter` function first forwards the `GXDefaultPrinter` message so that the standard view devices can be added, then adds two of its own: a 144 dpi 1-bit view device and a 144 dpi 8-color view device. This function uses a `for` loop to assign color values that correspond to the eight basic RGB colors. These values are shown in Table 3-5.

Table 3-5 Color values for an eight-color view device for the ImageWriter II printer

Color name	Color index	Red value	Green value	Blue value
White	0	0xFFFF	0xFFFF	0xFFFF
Yellow	1	0xFFFF	0xFFFF	0x0000
Magenta	2	0xFFFF	0x0000	0xFFFF
Red	3	0xFFFF	0x0000	0x0000
Cyan	4	0x0000	0xFFFF	0xFFFF
Green	5	0x0000	0xFFFF	0x0000
Blue	6	0x0000	0x0000	0xFFFF
Black	7	0x0000	0x0000	0x0000

The `SD_DefaultPrinter` function calls two local functions, the code for which is found in the QuickDraw GX sample code:

- The NewDeviceResolutionViewDevice function creates a view device object and sets up its mapping for 144 dpi.
- The PrinterHasColorRibbon function returns true if the desktop printer information says that the printing device has a color ribbon installed on it.

Establishing the Preferred Printing Characteristics

QuickDraw GX sends the GXDefaultJob message to initialize a new job object. You can override this message, which is described on page 4-47 in the chapter "Printing Messages," to add collections to the job or to establish printing preferences for the job.

The ImageWriter II printer driver overrides the GXDefaultJob message to add an item to the job collection that stores the preferred printing resolution and to initialize that preference to the highest resolution possible. This message override, the SD_DefaultJob function, is shown in Listing 3-6.

Listing 3-6 Establishing the printing resolution for the ImageWriter II printer

```
OSErr SD_DefaultJob()
{
   OSErr anErr;

   anErr = Forward_GXDefaultJob();
   if (anErr == noErr)
      {
      long  imagewriterOptions = kSuperRes;
      /* add high resolution preference item */
      anErr = AddCollectionItem(GXGetJobCollection(GXGetJob()),
               DriverCreator, 0, sizeof(imagewriterOptions),
               &imagewriterOptions);
      }
   return(anErr);
}
```

The SD_DefaultJob function adds an item to the job collection. Other printer drivers override the GXDefaultJob message for similar purposes. For example, one vector printer driver overrides this message to create a number of collection items that are used in its other functions. These items, which are added to the job collection, include the current carousel list, pen list, pens dialog box settings, and plot-quality settings.

When a user chooses the Page Setup dialog box, QuickDraw GX sends the GXJobDefaultFormatDialog message, which you can override to modify the contents of the dialog box. The ImageWriter II printer driver override of this message, SD_JobFormatDialog, determines if the job includes support for using the text-mode printing capabilities of the ImageWriter II, as shown in Listing 3-7. If the job does support text-mode printing, then text mode becomes the preferred printing mode.

The GXJobDefaultFormatDialog message is described on page 4-82 in the chapter “Printing Messages.”

Listing 3-7 Determining the preferred job-formatting mode

```
OSErr SD_JobFormatDialog(gxDialogResult   *theResult)
{
   OSErr                      anErr;
   gxJobFormatModeTableHdl    theJobFormatModeList;
   long                       i;
   gxJob                      theJob = GXGetJob();

   /* set up the job format mode information */

   anErr = GXGetAvailableJobFormatModes(&theJobFormatModeList);
   if ((!anErr) && (theJobFormatModeList))
      {
      for (i = 0; i <= (*theJobFormatModeList)->numModes - 1; ++i)
         {
         if ((*theJobFormatModeList)->modes[i] ==
                                            gxTextJobFormatMode)
            {
            GXSetPreferredJobFormatMode(gxTextJobFormatMode,
                                                     false);
            break;
            }
         }
      DisposHandle((Handle)theJobFormatModeList);
      }

   /*do normal dialogs after handling job format mode stuff */
   return(Forward_GXJobDefaultFormatDialog(theResult));

}
```

The SD_JobFormatDialog function calls the GXGetAvailableJobFormatModes function to determine which formatting modes the application supports. If text formatting mode is supported, SD_JobFormatDialog makes that the preferred 0mode by calling the GXSetPreferredJobFormatMode function. The GXGetAvailableJobFormatModes function is described on page 5-30 and the GXSetPreferredJobFormatMode function is described on page 5-30 in the chapter “Printing Functions for Message Overrides.”

Storing the Current Printer Configuration

QuickDraw GX sends the `GXOpenConnection` message to open a connection with a printing device. You can override this message, which is described on page 4-131 in the chapter "Printing Messages," to perform any special operations that you need to do at the time of connection.

The ImageWriter II printer driver overrides the `GXOpenConnection` message to update the printer configuration that is stored with the desktop printer. It first sends a query to the printing device for its hardware configuration, which includes information about the sheet-feeder and color-ribbon options. It then stores that information with the desktop printer. This message override, the `SD_OpenConnection` function, is shown in Listing 3-8.

Listing 3-8 Opening the connection with the printing device

```
OSErr SD_OpenConnection(void)
{
   OSErr    anErr;

   /* first, open the connection the standard way */
   anErr = Forward_GXOpenConnection();
   nrequire(anErr, OpenConnection);

   /*
      then, bring the desktop-printer configuration information
      up to date
   */
   anErr = UpdateConfiguration();
   nrequire(anErr, UpdateConfiguration);

   return(noErr);

/* exception handling */
UpdateConfiguration:
   GXCleanupOpenConnection();
OpenConnection:

   return(anErr);
}
```

The `SD_OpenConnection` function queries and stores the hardware configuration by calling a local function, `UpdateConfiguration`. This function, which is shown in Listing 3-9, calls the `FetchStatusString` function to query the ImageWriter II and then stores the information in the printer configuration file.

Listing 3-9 Getting information about the configuration of the printing device

```
OSErr UpdateConfiguration(void)
{
   Str32                        deviceName;
   OSErr                        anErr = noErr;
   ImageWriterConfigHandle      configHandle;
   ImageWriterConfigPtr         configPtr;
   Boolean                      isImageWriterII = false;
   ResType                      commType;


   /* find out printer name and how the printer is connected */
   GXGetPrinterName(GXGetJobOutputPrinter(GXGetJob()),
                                                    deviceName);
   anErr = GXFetchDTPData(deviceName, gxDeviceCommunicationsType,
               gxDeviceCommunicationsID, (Handle*)&configHandle);
   nrequire(anErr, FetchCommType);
   commType = **(ResType**)configHandle;
   DisposHandle((Handle) configHandle);

   /* store the communications type for future use */
   {
   SpecGlobalsHdl hGlobals = GetMessageHandlerInstanceContext();

   (**hGlobals).commType = commType;
   }

   /* find out the original configuration */
   if (GXFetchDTPData(deviceName, kImageWriterConfigType,
         kImageWriterConfigID, (Handle*)&configHandle) == noErr)
      {
      /* remember that it was an IW2 */
      configPtr = *configHandle;
      isImageWriterII = configPtr->isImageWriterII;
      DisposeHandle((Handle) configHandle);

      /*
         If this is not an ImageWriter II, bail out now
         because the timeout takes two minutes!
      */
      if (!isImageWriterII)
         return(noErr);
      }
```

```
else
   {

   /* if not sure yet, assume IW2 for PAP and IW1 otherwise */
   if (commType == 'PPTL')
      isImageWriterII = true;
   }

/* make a handle to hold configuration info for the printer */
configHandle = (ImageWriterConfigHandle)
                  NewHandle(sizeof(ImageWriterConfigRecord) );
anErr = MemError();
nrequire(anErr, NewHandle);

/* set up the default for the device */
configPtr = *configHandle;
configPtr->hasColorRibbon = false;
configPtr->hasSheetFeeder = false;
configPtr->isImageWriterII = true;

{
short    statusReturn;

/* query the device */
anErr = FetchStatusString(&statusReturn, (commType == 'PPTL'));

/*
   Scan the status string looking for information about
   printer kind and options.
*/
configPtr = *configHandle;
if ( anErr == gxAioTimeout )
   {
   /*
      If you don’t know what kind of printer it is and the
      request timed out, assume that it is an IW1.
   */
   if (!isImageWriterII)
      {
      anErr = noErr;
      configPtr->isImageWriterII = false;
      }
   }
   else
      {
      configPtr->hasColorRibbon =
               (statusReturn & (0x1000 >> kColorRibbonBit)) != 0;
      configPtr->hasSheetFeeder =
               (statusReturn & (0x1000 >> kSheetFeederBit)) != 0;
      }
   nrequire(anErr, FetchStatusString);
   }

   /* write out the new configuration */
   anErr = GXWriteDTPData(deviceName, kImageWriterConfigType,
                     kImageWriterConfigID, (Handle)configHandle);


/* exception handling */
FetchStatusString:
   DisposHandle((Handle) configHandle);

NewHandle:
FetchCommType:
   return(anErr);

}
```

The `UpdateConfiguration` function first calls the `GXFetchDTPData` function to access the communications type that is stored for the printer with the desktop printer. `UpdateConfiguration` next calls `GXFetchDTPData` to determine with which kind of printer it is communicating. Then, `UpdateConfiguration` calls the `FetchStatusString` function. This is a local function that sends a query to the hardware device and receives a status string back from it. `UpdateConfiguration` examines the status string to determine whether the printer has a sheet feeder or a color ribbon. Finally, `UpdateConfiguration` calls the `GXWriteDTPData` function to store the information that it has updated in the desktop printer.

Desktop printers are described in *Inside Macintosh: QuickDraw GX Printing*. The `GXFetchDTPData` function is described on page 5-27 and the `GXWriteDTPData` function is described on page 5-26 in the chapter “Printing Functions for Message Overrides.”

Responding to Direct-Mode Queries

QuickDraw GX sends the `GXJobFormatModeQuery` message to get or set format-mode information. This message only applies to direct-mode printing, which the user can select for printing to a specific printing device such as the ImageWriter II. Direct mode allows for accelerated printing of text for printing devices such as the ImageWriter II; however, illustrations cannot be printed while text direct-mode printing is selected.

QuickDraw GX can make various requests through the `GXJobFormatModeQuery` message that you need to respond to, including requests

- to get a list of the fonts that are supported for the job
- to get a list of the styles that are supported for the job
- to get the positioning constraints of the job for font sizes and for line drawing
- to set the current font style, which can be chosen from the list of supported styles

These requests apply only to text and line printing mode. The `GXJobFormatModeQuery` message is described on page 4-59 in the chapter "Printing Messages." For a complete list and description of the `GXJobFormatModeQuery` request types, see the chapter "Advanced Printing Features" in *Inside Macintosh: QuickDraw GX Printing.*

The ImageWriter II printer driver overrides the `GXJobFormatModeQuery` message to implement its responses to the queries. Its implementation is the `SD_JobFormatModeQuery` function, a portion of which is shown in Listing 3-10.

Listing 3-10 Responding to a query about the job format mode

```
OSErr SD_JobFormatModeQuery( gxQueryType theQuery,
                        void* srcData, void* dstData)
{
   OSErr    anErr = noErr;
   Handle   theFonts;
   Handle   theStyles;

   switch(theQuery)
   {
      case gxSetStyleJobFormatCommonStyleQuery:
      {
         char        *pStyleName;
         /* fetch the list of supported styles */
         anErr = Send_GXFetchTaggedDriverData('STR#',
                              kFormatModeStylesID, &theStyles);
         require(anErr == noErr, FailedToLoadStyles1);
         HNoPurge(theStyles);
         HLock(theStyles);

         /*
            Determine which style is being referenced and set
            the corresponding style (only two styles are
            currently supported).
         */

         if (**((short **) theStyles) == 2)
         {  /* the correct number of styles are in place */
            char  whichFace = 0;

            pStyleName = ((char *) *theStyles) + sizeof(short);

            if ( IUCompString(pStyleName, (char *) srcData) == 0 )
            {  /* user wants bold face */
               whichFace = bold;
            }
            else
            {  /* point to next name in the list */
               pStyleName += *pStyleName + 1;
               if ( IUCompString(pStyleName,
                                 (char *) srcData) == 0 )
               {  /* user wants the underline face */
                  whichFace = underline;
               }
            }

               /* if user specified a valid face, set it now */
            if (whichFace != 0)
            {
               SetStyleCommonFace((gxStyle) dstData,
               GetStyleCommonFace((gxStyle) dstData) | whichFace);
            }
         }
         /* else - something is wrong with our resource */

         DisposHandle(theStyles);   /* dump temporary handle */

         break;
      }
```

The `SD_JobFormatModeQuery` function is designed as a `switch` statement, with code similar to that of Listing 3-10 (for the `gxSetStyleJobFormatCommonStyleQuery` request) included for each of the queries. The full version of this function is found in the QuickDraw GX sample code.

Setting Up the Parameters for Printing

QuickDraw GX sends the GXSetupImageData message so that you can initialize any constant data that your driver needs to use for imaging a document. The ImageWriter II driver overrides this message with the SD_SetupImageData function, which sets up the following data for a print job:

- the draft character table for printing in text mode
- halftone data for printing in graphics mode
- the start-of-page string for unidirectional or bidirectional printing, depending on the print quality of the print job
- the raster packaging information, which is different for different qualities of printing
- print-quality information
- color-printing information

The GXSetupImageData message is described on page 4-92 in the chapter "Printing Messages." A portion of the SD_SetupImageData function is shown in Listing 3-11. The complete version of this function is found in the QuickDraw GX sample code.

Listing 3-11 Setting up the constant data for the print job

```
OSErr SD_SetupImageData( gxRasterImageDataHdl hImageData )
{

   OSErr                anErr;
   gxRasterImageDataPtr pImageData;
   Boolean              isJobNotFinalQuality, isTextJobFormatMode;
   long                 imagewriterOptions;

   /* do the default setup */
   anErr = Forward_GXSetupImageData(hImageData);
   nrequire(anErr, Forward_GXSetupImageData);

   /* test for 'final' quality mode */
   isJobNotFinalQuality = !JobIsBest(&imagewriterOptions);

   /* test for gxTextJobFormatMode */
   isTextJobFormatMode = ( GXGetJobFormatMode( GXGetJob() ) ==
                                              gxTextJobFormatMode);

   /*
      If the job is not final quality and is not using text mode,
      downgrade the imaging data to lower quality.
   */
   if (isJobNotFinalQuality  ||  isTextJobFormatMode)
      {
      /* rough or text mode */
      pImageData = *hImageData;  /* dereference for size+speed */

      /* image at 80 or 72 dpi */
      if (imagewriterOptions & kSuperRes)
         pImageData->hImageRes = ff(80);
      else
         pImageData->hImageRes = ff(72);
      pImageData->vImageRes = ff(72);

      /*
         If using text mode, load the draft table;
         otherwise, set up the halftone data.
      */
      if (isTextJobFormatMode)
         {  /* load the draft table */
         Handle         draftTable;
         SpecGlobalsHdl hGlobals =
                              GetMessageHandlerInstanceContext();

         anErr = Send_GXFetchTaggedDriverData('idft',
                          gxPrintingDriverBaseID, &draftTable);
         nrequire(anErr, FailedToLoadDraftTable);

         (**hGlobals).draftTable = draftTable;
         }
      else
         {
         /*
            Use dither level that looks better at 72 dpi
            than do the default values from the resources.
         */
         pImageData->theSetup.planeSetup[0].planeHalftone.method
                                                          = 4;
         /* turn off color matching when in non-final mode */
         pImageData->theSetup.planeSetup[0].planeProfile = nil;
         }
...
```

The remainder of the `SD_SetupImageData` function operates similarly, modifying the default values for varying resolution values.

The `JobIsBest` function that is called by `SD_SetupImageData` accesses the job collection to establish the print quality. The code for this function is shown in Listing 3-12. The quality information is stored as an item in the job collection that specifies how the user wants the job printed. This information is accessed from the job collection using the `gxQualityTag` tag ID.

Listing 3-12 Establishing the print quality

```
Boolean JobIsBest(long *imagewriterOptions)
{
   Boolean        isFinal;
   gxQualityInfo  jobQualitySettings;
   long           itemSize = sizeof(jobQualitySettings);
   OSErr          status;
   Collection     jobCollection;

   /* cache the collection */
   jobCollection = GXGetJobCollection(GXGetJob());

   isFinal = false;
   status = GetCollectionItem(jobCollection, gxQualityTag,
               gxPrintingTagID, &itemSize, &jobQualitySettings);

   if ( (status == noErr) && (jobQualitySettings.currentQuality
                     == (jobQualitySettings.qualityCount-1)) )
      isFinal = true;

   /* default is kSuperRes */
   *imagewriterOptions = kSuperRes;
   itemSize = sizeof(imagewriterOptions);
   status = GetCollectionItem(jobCollection, DriverCreator, 0,
                              &itemSize, imagewriterOptions);

   return(isFinal);

}
```

The `JobIsBest` function returns `true` if the current print job is printed in final quality; otherwise, it returns `false`. It also accesses and returns the ImageWriter II options from the job collection.

Managing Special Page Handling

QuickDraw GX sends the `GXStartSendPage` message during imaging, just before each page is rendered. The ImageWriter II printer driver overrides this message with the `SD_StartSendPage` function to manage manual feeding of paper. The `SD_StartSendPage` function first checks to see if the entire print job uses automatic paper feeding; if not, it determines if the paper type of the page that is about to be rendered is a manually fed paper type. If so, `SD_StartSendPage` displays a printing alert box, as described in the section "Displaying Status Information and the Printing Alert Boxes" beginning on page 3-41.

The first portion of the `SD_StartSendPage` function uses the job collection to determine if the entire print job uses automatic paper feeding, as shown in Listing 3-13. The `GXStartSendPage` message is described on page 4-136 in the chapter "Printing Messages."

Listing 3-13 Determining if the print job uses any manually fed pages

```
{
   OSErr              anErr = noErr;
   gxJob              theJob = GXGetJob();
   Collection         jobCollection;
   gxPaperFeedInfo    paperFeed;
   long               itemSize = sizeof(paperFeed);
   ResType            commType;
   short              statusReturn;

   jobCollection = GetJobCollection(theJob);

   /* cache the communications type */
   commType = (**(SpecGlobalsHdl)
                 GetMessageHandlerInstanceContext()).commType;
   if (commType == 'PPTL')
      {
      FetchStatusString(&statusReturn, true, true);
      nrequire(anErr, FetchStatusString);
      }
   else
      statusReturn = 0;

   /* first, determine if the entire job is auto feed */
   paperFeed.autoFeed = true;
   (void) GetCollectionItem(jobCollection, gxPaperFeedTag,
                            gxPrintingTagID, &itemSize, &paperFeed);
/*
```

```
   If the entire job is not auto feed, the value of the
   paperFeed.autoFeed field will now be changed to false.
*/
   .....
```

If the `GetCollectionItem` call succeeds in finding a paper-feed item, then the value of the `paperFeed.autoFeed` field is determined by the retrieved value. Otherwise, the value of the `paperFeed.autoFeed` field remains `true`, as it was before making the call. If this field is `false`, the job includes manual feeding.

The next portion of the `SD_StartSendPage` function executes if the print job includes any manually fed pages. This portion loops through the list of manual-feed paper-type names and uses the job collection to determine if the entire job uses automatic paper feeding, as shown in Listing 3-14.

Listing 3-14 Finding the manual-feed paper name

```
if (!paperFeed.autoFeed)
   {
   /* get the manual-feed list, using the paper-feed size */
   gxManualFeedInfo   **feedHandle;

   feedHandle = (gxManualFeedInfo**) NewHandle(0);
   anErr = MemError();
   nrequire(anErr, FailedNewHandle);

   anErr = GetCollectionItemHdl(jobCollection, gxManualFeedTag,
                           gxPrintingTagID,(Handle)feedHandle);
   if (anErr == noErr)
      {
      Str31             paperName;
      short             idx;
      gxManualFeedInfo   *pFeed;

      /* get name of this page's paper type */
      GXGetPaperTypeName(GXGetFormatPaperType(pageFormat),
                                    paperName);

      paperFeed.autoFeed = true;

      /* lock and dereference for the loop */
      HLockHi((Handle) feedHandle);
      pFeed = *feedHandle;
      /* look for the manualy fed paper-type name */
      for (idx = 0; idx < pFeed->numPaperTypeNames; ++idx)
         {
         Ptr pName = &pFeed->paperTypeNames[idx];

         if (IUMagIDString( paperName, pName,
                            paperName[0], *pName+1) == 0)
            {
            paperFeed.autoFeed = false;
            break;
            }
         }
      }

   DisposHandle((Handle) feedHandle);
   FailedNewHandle:
      ;
   }
```

If the manual-feed paper-type name is found (if `paperFeed.autoFeed` is `false` after exiting the loop) or if the paper-name list is empty, then the remainder of `SD_StartSendPage` displays a printing alert box to tell the user to manually feed the appropriate paper. This portion of the `SD_StartSendPage` function is shown in the next section. The code for the entire `SD_StartSendPage` function is found in the QuickDraw GX sample code.

Displaying Status Information and the Printing Alert Boxes

You can report status information to the user during the printing process to keep the user informed. Informative text about the printing status is displayed in the desktop printer window.

In addition, when the user has to perform an action before printing can continue, your driver may need to display a printing alert box. You process a printing alert following this sequence of actions:

1. Make a status record that contains the request to the user.
2. Call the `GXAlertTheUser` function to display the status information in a printing alert box. The `GXAlertTheUser` function is described on page 5-18 in the chapter “Printing Functions for Message Overrides.”
3. Keep sending the printing alert (repeatedly call `GXAlertTheUser`) until one of three actions occurs:
   - The condition resolves itself.
   - The user responds by clicking a button in the printing alert box.
   - An error of some sort occurs.

4. If necessary, call `GXAlertTheUser` again with other informational text so that the user can dismiss the printing alert box. You need to do this if an error occurred or if the problem resolved itself. If the user dismisses the printing alert box, you don't need to take this step.

In the ImageWriter II printer driver, the `SD_StartSendPage` function displays a printing alert box when the user has to manually feed a page into the printing device, as shown in Listing 3-15.

Listing 3-15 Displaying a printing alert box with printer status information

```
{
      StatusRecordPtr      pStat;

   /* make a status record with the request to the user */
   pStat = (StatusRecordPtr) NewPtrClear(sizeof(gxStatusRecord) +
                                 sizeof(gxManualFeedRecord));
   anErr = MemError();
   nrequire(anErr, NewPtrClear);
   /* use the built-in status resource for this */
   pStat->statResId = gxUnivAlertStatusResourceId;
   pStat->dialogResult = nil;

   if (!paperFeed.AutoFeed)
      {
      gxManualFeedRecord *pFeed;

      /* use the provided manual-feed alert text string */
      pStat->statResIndex = gxUnivManualFeedIndex;
      pStat->bufferLen  = sizeof(gxManualFeedRecord);
      pFeed = (gxManualFeedRecord*)&pStat->statusBuffer;
      /* the user can choose to switch to auto feed */
      pFeed->canAutoFeed = true;
      GXGetPaperTypeName(GXGetFormatPaperType(pageFormat),
                                 pFeed->paperTypeName);
      }
   else
      {
      gxOutOfPaperRecord *pOut;

      pStat->statResIndex = gxUniveOutOfPaperIndex;
      pStat->bufferLen = sizeof(gxOutOfPaperRecord);
      pOut = (gxOutofPaperRecord *)&pStat->statusBuffer;
      GXGetPaperTypeName(GXGetFormatPaperType(pageFormat,
                                       pOut->paperTypeName);
      }

   do /* loop, sending the alert until it gets resolved */
      {
      anErr = GXAlertTheUser(pStat);

      /* if the paper suddenly got loaded, do an OK */
      if (commType == 'PPTL')
         {
         FetchStatusString(&statusReturn, true);
         if ((statusReturn & kOutOfPaperMask) == 0)
            {
            pStat->dialogResult = ok;
            anErr = noErr;
            }
         }
      } while ((anErr == noErr) && (pStat->dialogResult == nil));

      /* decide what to do, based on the user’s response */
   switch ( pStat->dialogResult )
      {
      case ok:                /* the paper is now loaded */
         break;

      case cancel:            /* user wants to stop printing */
         anErr = gxPrUserAbortErr;
         break;

      case gxAutoFeedButtonId:/* do rest of job with auto feed */
         paperFeed.gxAutoFeed = true;
         (void) AddCollectionItem(jobCollection, gxPaperFeedTag,
                           gxPrintingTagID, itemSize, &paperFeed);
         break;
      }

   DisposPtr((Ptr) pStat); /* done with status now, so dispose */
   }
/* now display the "Sending data to the printer" text */
if (anErr == noErr)
   anErr = GXReportStatus(kDriverStatus, kSendingData);
...
}
```

This printing alert box makes use of one of the built-in alert conditions that are defined for printer drivers. Table 3-6 lists the predefined alert conditions, each of which is an index into the corresponding informational text strings stored in a predefined printing alert (`'plrt'`) resource.

Table 3-6 Predefined alert conditions for printing device drivers

Constant	Value	Explanation
`gxUnivManualFeedIndex`	2	Paper needs to be manually fed into the printing device
`gxUnivFailToPrintIndex`	3	The printing device could not print the job
`gxUnivPaperJamIndex`	4	A paper jam has occurred on the printing device
`gxUnivOutOfPaperIndex`	5	The printing device is out of paper
`gxUnivNoPaperTrayIndex`	6	The required paper tray is not in place
`gxUnivPrinterReadyIndex`	7	The printing device is ready to print

You can add your own informational text to use with the `GXAlertTheUser` function by defining printing alert (`'plrt'`) resources, which are described in the chapter “Printing Resources” in this book.

Checking for When an Alert Condition Resolves Itself

Listing 3-15 on page 3-42 includes a loop that continues to call `GXAlertTheUser` to display the printing alert box until the user feeds the paper. If you are using a printing device that can automatically detect paper feeding, you could change the loop as shown in Listing 3-16.

Listing 3-16 Checking if an alert condition has resolved itself

```
do
{
   anErr = GXAlertTheUser(pStatus);
   /* see if the user has loaded the paper */
   if ((anErr == noErr) && (pStatus->dialogResult ==nil))
      {
         Boolean userFedPaper;

         userFedPaper = CheckToSeeIfPaperWasFed();
         if (userFedPaper)
            {
```

```
            pStatus->statResIndex = gxUnivPrinterReadyIndex;
            anErr = GXAlertTheUser(pStatus);
            pStatus->dialogResult = ok;
            }
      }
} while ((pStatus->dialogResult == nil) && (anErr == noErr));
```

In this version of the alert loop, the driver calls the `CheckToSeeIfPaperWasFed` function, which detects if the user has fed a page of paper into the printing device. When this happens, the manual-feed text string in the printing alert box is replaced with the string "Printer is ready," which the user can hide at any time.

Filling In Alert Information at Run Time

When you display a printing alert box and you need to dynamically fill in parts of the alert text string, you have to override two messages:

- Override the `GXInitializeStatusAlert` message to fill in the printing alert box at run time. This message is described on page 4-164 in the chapter "Printing Messages."
- Override the `GXHandleAlertEvent` message to manage what happens when an event occurs in the printing alert box. This message is described on page 4-165 in the chapter "Printing Messages."

You can also override the `GXHandleAlertFilter` message to work with any controls that have been installed in the printing alert box. This message is described on page 4-168 in the chapter "Printing Messages."

Listing 3-17 shows portions of the overrides for these messages that could be used to tell the user that a new pen carousel needs to be placed in a plotter.

Listing 3-17 Modifying alert information at run time

```
OSErr MyInitializeStatusAlert( StatusRecordPtr pStatus,
                                 DialogPtr *pDialog )
{
   OSErr anErr = noErr;

   /* first see if this message is for your driver */
   if (pStatus->statusOwner == kDrvrCreatorType)
      {
      if (pStatus->statusId == kChangeCarouselsDlogID)
         {
         *pDialog = GetNewDialog( kChangeCarouselsDlogID, nil,
                                 (WindowPtr)-1);
         if (*pDialog == nil)
            anErr = resNotFound;
         else     /* fill in run-time information */
            FillInDialogStrings( pStatus, pDialog );
         }
      else...     /* handle other status conditions */

      }
   else           /* the status text belongs to someone else */
      Forward_GXInitializeStatusAlert( pStatus, pDialog );

   return( anErr );
}


OSErr MyHandleAlertEvent( StatusRecordPtr pStatus, DialogPtr
              *pDialog, EventRecord *theEvent, short *itemHit )
{
   OSErr anErr = noErr;

   /* first see if this message is for your driver */
   if (pStatus->statusOwner == kDrvrCreatorType)
      {
      if (pStatus->statusId == kChangeCarouselsDlogID)
         {
         HandleCarouselDialogEvent( pDialog, theEvent, itemHit );
         pStatus->dialogResult = *itemHit;
         }
      else.../* handle other status condition events */

      }
   else     /* the status message belongs to someone else */
      Forward_GXHandleAlertEvent( pStatus, pDialog, theEvent,
                                                  itemHit );

   return( anErr );
}
```

Each of these functions first checks if the status condition is one that it handles. If not, the condition is passed on to the next handler in the message chain. If the condition is one that the driver handles, the function performs the needed operation: either filling in the name of the carousel that needs to be displayed in the printing alert box or handling an event in the alert box. If the driver handles the alert in these functions, it does not forward the message to the other message handlers.

Displaying Status Text During Printing

When you want to display status information to the user and it does not require a response, you can call the `GXReportStatus` function, providing it with the ID of a status resource and the ID of the static text in that resource that you want sent to the desktop printer window.

QuickDraw GX provides several status (`'stat'`) resources with a number of predefined text strings that you can display to the user. The printer-status text strings, which are listed in Table 3-7, are located in the page transmission status resource. Some of these strings are repeated, with one entry associated with an alert condition, and the other used to display informational status.

Table 3-7 Status text IDs in the page transmission status resource

Text ID	Text string
1	"Waiting for the next sheet of paper…"
2	"Sending part of the page…"
3	"Preparing part of the page…"
4	"The printer is out of paper."
5	"The printer has a paper jam."
6	"The paper tray is improperly loaded."
7	"The printer door is open. Please close it." (alert version)
8	"The printer door is open. Please close it." (status version)
9	"A print test is in progress. Please wait…"
10	"The printer fixing unit is being heated. Please wait…"
11	"The toner cartridge is improperly loaded." (alert version)
12	"The toner cartridge is improperly loaded." (status version)
13	"Printing the page."
14	"Trying to locate the printer…"
15	"Opening a connection to the printer…"

Rendering the Page on the Printing Device

QuickDraw GX sends the `GXRenderPage` message once for each page, when the QuickDraw GX shape that is the picture of the page needs to be translated into a stream of data that the printing device can use to render the output page. QuickDraw GX's default implementation for this function handles most of the work for you. For a raster printing device, it converts the page into bitmap data and sends the `GXRasterPackageBitmap` message before sending the bitmap to the printing device.

When the raster printing-device head has to be moved down the page, QuickDraw GX sends the `GXRasterLineFeed` message.

Printing is accomplished by adding bytes of data to a buffer that you send to the printing device with the `GXBufferData` message.

The ImageWriter II printer driver overrides the `GXRenderPage` message to add its own operations. Its override function, `SD_RenderPage`, determines if the page is to be printed in text mode or not. If so, it calls the local function `PrintPageInDraftMode` to do the printing. If the page is printed in graphics mode, `SD_RenderPage` sends escape sequences to the printing device to set up the page size and then forwards the `GXRenderPage` message so that the default implementation can manage the page printing. The `GXRenderPage` message is described on page 4-96 in the chapter "Printing Messages."

The ImageWriter II printer driver overrides the `GXRasterLineFeed` message with the `SD_LineFeed` function. This function adjusts the line-feed size if the driver is printing at low resolution because the ImageWriter II assumes that line-feed commands are expressed in high-resolution values. `SD_LineFeed` forwards the `GXRasterLineFeed` message to allow the default implementation to send the appropriate escape sequences to the printing device. The `GXRasterLineFeed` message is described on page 4-98 in the chapter "Printing Messages."

The ImageWriter II printer driver also overrides the `GXRasterPackageBitmap` message. The `SD_PackageBitmap` override function fills in a buffer with the raster data, applying ImageWriter II run-length encoding and rotating the data. The `GXRasterPackageBitmap` message is described on page 4-100 in the chapter "Printing Messages."

The `SD_RenderPage`, `SD_LineFeed`, and `SD_PackageBitmap` override functions are shown in the QuickDraw GX sample code.

Terminating the Print Job

At the end of printing, your printer driver needs to clean up any storage that it has allocated. You can override the `GXShutDown` message to handle this task. The ImageWriter II printer driver overrides the `GXShutDown` message with the `SD_Shutdown` function, which is shown in Listing 3-18. The `GXShutDown` message is described on page 4-44 in the chapter "Printing Messages."

Listing 3-18 Terminating the print job

```
OSErr SD_ShutDown(void)
/*
   GXShutDown is called when the printing job is done. A good
   thing to do is to get rid of any additional storage.
*/
{
   /* clean up the globals */
   SpecGlobalsHdl hGlobals = GetMessageHandlerInstanceContext();

   /* get rid of the draft table if you allocated it */
   DisposHandle((**hGlobals).draftTable);

   /* get get rid of allocated storage */
   DisposHandle((Handle) hGlobals);

   /* clear out the globals - to avoid double disposes */
   SetMessageHandlerInstanceContext(nil);

   return(noErr);
}
```

The `SD_ShutDown` function essentially reverses what the `SD_Initialize` function did at the start of printing, as shown in the section "Initializing the ImageWriter II Environment" beginning on page 3-24.

Providing Compatibility in the ImageWriter II Driver

This section describes the override functions that comprise the portion of the ImageWriter II printer driver that provides compatibility with the Macintosh Printing Manager. This portion of the driver needs to override any of the Printing Manager compatibility messages that must take into account specific characteristics of the printing device. The functions that override such messages include the following:

- The `SD_ConvertPrintRecordTo` function uses knowledge of the ImageWriter printer characteristics to convert an old print record into a universal print structure.
- The `SD_ConvertPrintRecordFrom` function converts a universal print structure into Macintosh Printing Manager print record for the ImageWriter II.
- The `SD_PrintRecordToJob` function forwards the `GXPrintRecordToJob` message and then converts one of the resultant settings into an option in the job collection that is used by the driver.
- The `SD_PrValidate` function validates the current print record by filling it in with the current settings for the application and printing device.
- The `SD_PrJobInit` function first forwards the `GXPrJobInit` message to display the default settings in the Print dialog box, then disables some of the items, based on the current settings for the printing device.

All of the Macintosh Printing Manager compatibility messages, including those that correspond to these override functions, are described in the section "Compatibility Messages" beginning on page 4-147 in the chapter "Printing Messages." The code for the overrides used by the ImageWriter II printer driver is found in the QuickDraw GX sample code.

Color Printing

Printing in color is greatly simplified with QuickDraw GX. All of the color-processing information that you need to provide is specified in resources, which means that you don't need to override any messages or call any functions for color printing to work properly.

For raster printing devices, you specify color information in the raster driver preferences (`'rdip'`) resource, which defines the color space, color profile, and other color information for each color plane on the printing device. For PostScript printing devices, you define the PostScript color space in the PostScript preferences (`'pdip'`) resource. Both of these resources are described in the chapter "Printer Resources" in this book.

If you want to modify the color information at run time, you can override the `GXSetupImageData` message and modify the image data that QuickDraw GX created for your driver. The ImageWriter II printer driver uses the override function `SD_SetupImageData` to modify the halftone information when printing in graphics mode, as shown in Listing 3-11 on page 3-36. The `GXSetupImageData` message is described on page 4-92 in the chapter "Printing Messages." The data structures used to represent the imaging data for each imaging system are described in the chapter "Printing Messages" in this book.

For more information about color printing on PostScript printers, see the *PostScript Language Reference Manual*, 2nd Edition.

Color Matching

Macintosh system software includes ColorSync, which provides color matching for drivers and applications. Color matching is the process that allows different devices to display the same color, providing the user with an accurate color representation in a device-independent manner.

You can provide color matching in your QuickDraw GX printer drivers without overriding any messages or calling any functions. All that you need to do is provide at least one color profile (`'prof'`) resource in your driver. These resources, each of which describes how colors are represented on a specific device, are described in *Inside Macintosh: Advanced Color Imaging.* Although many printer drivers need only one color profile resource, you can define more than one such resource for your driver. Each profile can be associated with a specific page format or paper type. For example, the Apple Color Printer driver defines color profiles for three different media types that affect the appearance of color: coated paper, transparency film, and plain paper.

When you define a color profile resource for your printer driver, QuickDraw GX reads the resource data, creates a color profile object from the data, and associates that object with your printing device. If you want to create a color profile dynamically, you can override the `GXFetchTaggedData` message, which QuickDraw GX uses to read the resource data. The `GXFetchTaggedData` message is described on page page 4-45 in the chapter “Printing Messages.”

An application program can call the `GXFindPrinterProfile` function to query your printer driver. When an application calls this function, QuickDraw GX sends the corresponding `GXFindPrinterProfile` message. You can override this message to return to the application which profile you want to use. The application can then call the `GXSetPrinterProfile` function to change this information. Once again, QuickDraw GX sends the corresponding message—in this case, the `GXSetPrinterProfile` message. You can also override this message in your printer driver.

An application program can also call the `GXFindFormatProfile` function to query your printer driver to find out which color profile is used for a specific page format. When an application calls this function, QuickDraw GX sends the corresponding `GXFindFormatProfile` message. You can override this message to return to the application which profile you want to use. The application can then call the `GXSetFormatProfile` function to change this information. Once again, QuickDraw GX sends the corresponding message—in this case, the `GXSetFormatProfile` message. You can also override this message in your printer driver.

The `GXFindPrinterProfile`, `GXSetPrinterProfile`, `GXFindFormatProfile`, and `GXSetFormatProfile` message are described in the section “Color Profile Messages” beginning on page 4-62. The corresponding functions that cause these messages to be sent are described in *Inside Macintosh: QuickDraw GX Printing*.

If the application is changing color profiles on a per-format basis, you also need to override the `GXImagePage` message. One of the parameters for this message is a handle to imaging-system-specific data, which contains a handle to a color profile. In your override, you need to change which color profile the handle points to and forward message. The `GXImagePage` message is described on page 4-94 in the chapter “Printing Messages.”

Color Matching on PostScript Devices

If you are writing a printer driver for a PostScript device, you need to perform a few additional tasks to optimize color matching:

- Choose the color space your driver uses. The color space tells QuickDraw GX what kind of PostScript operators to use when specifying colors for your printing device.
- If your PostScript device supports Level 2 of the PostScript language, offload the color-matching work to the printing device rather than having the Macintosh perform the matching work (which can be time consuming).
- Generate portable PostScript code to guarantee that the best output results are generated for any PostScript device with which your driver is used.

The following sections discuss these three tasks.

Choosing a Color Space for a PostScript Printing Device

You must choose one of three color-space values for a PostScript device, each of which defines which PostScript operators QuickDraw GX uses to send color information. Table 3-8 shows the color-space values that you can use.

Table 3-8 PostScript color-space choices

Color space	**PostScript operators used by QuickDraw GX**
`gxRGBSpace`	`setrgbcolor` and `colorimage`
`gxCMYKSpace`	`setcmykcolor` and `colorimage`
`gxGraySpace`	`setgray` and `image`

The color-space values are described the chapter “Colors and Color-Related Objects” in *Inside Macintosh: QuickDraw GX Objects.* If you specify the wrong color space, then PostScript errors are generated. For example, if you specify the `gxCMYKSpace` color-space and your driver is connected to a printer that does not support the `setcmykcolor` operator, an error occurs. You can get around this situation by generating portable PostScript code, as described in the section “Generating Portable PostScript” on page 3-53.

Using PostScript’s Color Matching

If your printer driver is communicating with a PostScript device that supports Level 2 of the PostScript language, you can offload the color matching to the device rather than performing this time-consuming work in your driver. To do this, you need to specify two values in the PostScript preferences (`'pdip'`) resource for your driver: the language-level field must be set to 2, and you must include `gxUseLevel2ColorOption` in the render options field of the resource.

When you specify these values in your PostScript preferences resource, QuickDraw GX generates PostScript code that is optimized for the PostScript Level 2 interpreter. This means that QuickDraw GX allows the PostScript interpreter to perform the color matching rather than calling ColorSync to do so. QuickDraw GX converts the color space and color profile of the objects being printed into a Level 2 color-space dictionary by using the `setcolorspace` operator. The colors for the objects are set with the `setcolor` operator, and bitmaps are drawn with the Level 2 dictionary form of the `image` operator. If a bitmap’s color space is defined as 5 or 8 bits per component, the `image` operator is used at 8 bits per component. If a bitmap’s color space is defined as 10 bits per component, the `image` operator is used at 12 bits per component.

Since not all QuickDraw GX color spaces cannot be translated to PostScript Level 2, QuickDraw GX might need to convert the color space of the object. For example, if the object specifies `gxCIESpace`, which cannot be emulated with the `setcolorspace` operator, QuickDraw GX converts the colors into `gxXYZSpace`, which can be emulated.

If you set the language level to 2 but do not specify `gxUseLevel2ColorOption` as a rendering option in your PostSript preferences resource, QuickDraw GX generates code that is optimized for the Level 2 interpreter, but calls ColorSync to perform color matching.

Generating Portable PostScript

Since your PostScript driver can be used with a variety of printing devices, you need to be careful about which color space you specify. If you specify a color space that is not supported by the printing device, PostScript errors are generated. The best way to avoid this problem is to tell QuickDraw GX to generate portable PostScript code, which can be executed on any PostScript printing device and gives the best results that the printer can produce.

The only color space that is guaranteed to work on all PostScript devices is `gxGraySpace`. However, this color space generates grayscale even on a printer that supports color. To get QuickDraw GX to produce PostScript data that contains all of the color information but still renders on a monochrome device in grayscale, you need to specify two values in the PostScript preferences (`'pdip'`) resource: include `gxPortablePostScriptOption` in the render options field, and specify `gxRGBSpace` as the device's color-space value.

When you specify these values, QuickDraw GX defines PostScript procedures that emulate any color operators that are not available on the output device. QuickDraw GX also generates PostScript code to set up a Level 2 color space that is based on the color profile defined for your driver. If your driver is connected to a device that has Level 2 specified in its profile, color-matched output is generated.

Using Resources in Drivers

You need to provide a number of resources for drivers, just as you do for printing extensions. Some of these resources are required for all drivers, others are required for drivers that use a certain imaging system, and others are optional. All of the resources are described in the chapter "Printing Resources" in this book. Table 3-9 summarizes the resources that you can use in all printer drivers.

Table 3-9 Resource types used to define a printer driver

Resource type	Count	Description
`'over'`	1, 2, or 3	Defines which messages your driver needs to receive
`'isys'`	1	Specifies what kind of imaging system the driver uses
`'vers'`	1 or more	Specifies which version of QuickDraw GX the driver uses

continued

Table 3-9 Resource types used to define a printer driver (continued)

Resource type	Count	Description
'comm'	1 or more	Specifies printing-device communications parameters
'cust'	0 or 1	Specifies mappings from the Macintosh Printing Manager to the new driver architecture
'resl'	0 or 1	Specifies the horizontal and vertical resolution supported by a driver
'PREC'	0 or 1	Defines a default print record for your driver
'iobm'	0 or 1	Defines the timeout and buffering communications parameters for your driver
'cpts'	0 or 1	Specifies how your printing device can be removed and replaced on the network
'look'	1	Defines the type of communications that is used by the driver
'stat'	0 or more	Defines status messages for display
'plrt'	0 or more	Defines alert messages for display
'ppnl'	0 or more	Defines the contents of a panel that you are adding to a dialog box
'xdtl'	0 or more	Provides information that QuickDraw GX needs to execute panel controls
'ptyp'	0 or more	Defines characteristics of a paper type
'FREF'	1 or more	Specifies a file reference for the printer driver icons
'PACK'	1	Specifies Chooser package information
'LDEF'	1	Specifies Chooser list definition information
'BNDL'	1	Associates the printer driver with its icons and any files that it uses
'DLOG'	0 or more	Provides a dialog box template
'DITL'	0 or more	Contains an item list for a dialog box
'dctl'	0 or more	Specifies the dialog box control information for compatibility with Macintosh Printing Manager application dialog boxes
'stab'	0 or more	Defines a range of scaling for the scaling choice in the Print dialog box that is used for Macintosh Printing Manager compatibility
'ICN#'	1 or more	Defines a large (32-by-32 pixel) icon, 1-bit depth, with mask (if not provided, a generic driver icon is used)
'icl4'	0 or more	Defines a large icon, 4-bit depth (strongly recommended but not required)

Table 3-9 Resource types used to define a printer driver (continued)

Resource type	Count	Description
`'icl8'`	0 or more	Defines a large icon, 8-bit depth (strongly recommended but not required)
`'ics#'`	0 or more	Defines a small (16-by-16 pixel) icon, 1-bit depth, with mask (if not provided, a generic driver icon is used)
`'ics4'`	0 or more	Defines a small icon, 4-bit depth (strongly recommended but not required)
`'ics8'`	0 or more	Defines a small icon, 8-bit depth (strongly recommended but not required)

This section provides examples of several of these resources as used in the ImageWriter II printer driver. The contents of the the `oldapp.r` and `newapp.r` files, which contain the resource definitions for the ImageWriter II driver, are shown in the QuickDraw GX sample code.

You use some of the resources in Table 3-9 for user interface features, including the icons for your driver, resources that allow users to interact with your driver, and resources that the Chooser uses when the user selects your driver. The Finder interface resources, which are common to all Macintosh drivers, are described in *Inside Macintosh: More Macintosh Toolbox*. The icons that you must define for your driver when the user chooses it as the desktop printer are described in the section "Defining Desktop Printer Icons for Your Printer Driver" beginning on page 3-66.

All of the resources that you define for your printer drivers need to be loaded into the system heap and need to be purgeable. System resources are stored in the system heap as opposed to the application heap, where application resources are stored. Purgeable resources can be purged by the Memory Manager when space is required, as described in *Inside Macintosh: Memory*. You need to specify these attributes in the first line of every resource that you define for your driver, as is done in every resource example in this chapter.

Defining Code Segments in Your Driver

The code segments that you use to implement your printer driver must use segment type `'pdvr'`, which is defined by the constant `gxPrinterDriverType`. The ID of your first code segment needs to be 0, and you need to increment the ID by 1 for each subsequent segment in your driver.

Defining Version Compatibility for Your Printer Driver

Your printer driver must contain at least one version (`'vers'`) resource that defines its compatibility with QuickDraw GX. Version resources are used to record version information for Macintosh applications. You need to include a version resource with an ID of `gxPrintingDriverBaseID` that defines with which version of QuickDraw GX your driver is compatible. For the current version, the value of the first byte of the resource definition must be either 1 or 0. Listing 3-19 shows the version resource that defines QuickDraw GX compatibility for the ImageWriter II printer driver.

Listing 3-19 The QuickDraw GX version resource for the ImageWriter II printer driver

```
resource 'vers' (gxPrintingDriverBaseID, sysHeap, purgeable)
{
   0x01, 0x00, release, 0x00,
   verUS,
   "1.00",
   "1.00, Copyright \251 Apple Computer, Inc. 1989-1993"
};
```

You can also include standard version resources in the resource files for your printer driver. These resources are described in *Inside Macintosh: Macintosh Toolbox Essentials*.

Specifying Which Messages Your Driver Overrides

You must include an override (`'over'`) resource in your printer driver to provide QuickDraw GX with a list of the printing messages that your driver is overriding. Each entry in the override resource specifies a message and information about the resource in which the code segment for the message override is found. You need to include separate override resources for universal messages and for messages specific to an imaging system. The override resource is described on page 6-13 in the chapter “Printing Resources.”

The code segment information for each message includes the resource ID of the code segment and the offset into the code segment that contains the instruction to jump to the message-override function. The first 4 bytes of the code segment are reserved for use by QuickDraw GX and must be 0; thus, the first offset location is 4.

Each override resource in a printer driver has an ID of `gxPrintingDriverBaseID` for universal messages, an ID of `gxPrintingDriverBaseID+1` for messages specific to an imaging system, and an ID of `gxPrintingDriverBaseID+2` for Macintosh Printing Manager compatibility messages, as shown in Listing 3-20.

Listing 3-20 Override resources for the ImageWriter II printer driver

```
#define firstOffset  4
#define segmentID    NewSegID

resource gxOverrideType (gxPrintingDriverBaseID+0, sysHeap,
                                                   purgeable)
{
   {
   gxInitialize,              segmentID, firstOffset + 0,
   gxShutDown,                segmentID, firstOffset + 4,
   gxDefaultPrinter,          segmentID, firstOffset + 8,
   gxDefaultFormat,           segmentID, firstOffset + 12,
   gxDefaultJob,              segmentID, firstOffset + 16,
   gxJobDefaultFormatDialog,  segmentID, firstOffset + 20,
   gxJobFormatModeQuery,      segmentID, firstOffset + 24,
   gxRenderPage,              segmentID, firstOffset + 28,
   gxOpenConnection,          segmentID, firstOffset + 32,
   gxCloseConnection,         segmentID, firstOffset + 36,
   gxStartSendPage,           segmentID, firstOffset + 40,
   gxSetupImageData,          segmentID, firstOffset + 44,
   };
};

resource gxOverrideType (gxPrintingDriverBaseID + 1, sysHeap,
                                                      purgeable)
{
   {
   gxRasterPackageBitmap,segmentID, firstOffset + 48,
   gxRasterLineFeed,    segmentID,  firstOffset + 52,
   };
};

#define segmentID    OldSegID

resource gxOverrideType (gxPrintingDriverBaseID + 2, sysHeap,
                                                      purgeable)
{
   {
   gxConvertPrintRecordTo,  segmentID, firstOffset + 0,
   gxConvertPrintRecordFrom,segmentID, firstOffset + 4,
   gxPrintRecordToJob,      segmentID, firstOffset + 8,
   gxPrValidate,            segmentID, firstOffset + 12,
   gxPrJobInit,             segmentID, firstOffset + 16,
   };
};
```

Each resource in Listing 3-20 lists the messages that the ImageWriter II driver overrides. The first resource lists the universal messages, the second resource lists the messages specific to the raster imaging system, and the third resource lists the Macintosh Printing Manager compatibility messages. The code for the compatibility message overrides is located in a different code segment.

Each message listed in the resources specifies the name of the message, the ID of that segment, and where to find the message in the jump table. The ID of the code segment for all of these messages is defined by the constant `segmentID`. The jump instructions to the code that implements the overrides are each 4 bytes long. The first jump is found at the first offset location (byte 4), and each subsequent jump is found 4 bytes beyond the previous one.

Defining the Imaging System Type of Your Driver

You must include an imaging system (`'isys'`) resource in your printer driver to tell QuickDraw GX which imaging system your driver uses. The imaging system resource is described on page 6-33 in the chapter "Printing Resources." You must include exactly one of these resources, and it must contain one of the imaging system constants that are shown in Table 3-10.

Table 3-10 Imaging system values

Constant	Explanation
`gxRasterPrinterType`	The driver works with raster printing devices
`gxPostscriptPrinterType`	The driver works with PostScript printing devices
`gxVectorPrinterType`	The driver works with vector printing devices

The ImageWriter II printer is a raster printing device; thus, its imaging system resource specifies the `gxRasterPrinterType` value, as shown in Listing 3-21.

Listing 3-21 The imaging system resource for the ImageWriter II printer driver

```
resource gxImagingSystemSelectorType (gxImagingSystemSelectorID,
                                                sysHeap, purgeable)
{
   gxRasterPrinterType
};
```

Specifying How Your Driver Communicates With the Device

You need to specify how your printer driver communicates with the physical printing device that it is driving. You do this with two resource types: the communications (`'comm'`) resource and the buffering and input/output preferences (`'iobm'`) resource, known simply as the buffering resource. Each communications resource specifies the parameters for a particular kind of printing-device communications, as described in the section "The Communications ('comm') Resource" beginning on page 6-36 in the chapter

"Printing Resources." The buffering resource, which is described on page 6-61 in the chapter "Printing Resources," specifies buffer size and timing parameters.

There are different kinds of communications resources—one for each kind of printing-device communications that QuickDraw GX supports. Each of these resource types has its own format. Listing 3-22 shows the communications resources that are defined for the ImageWriter II printer driver, which supports PAP, serial, and PrinterShare communications connections.

Listing 3-22 Communications resources for the ImageWriter II printer driver

```
resource 'comm' (-4096, sysHeap, purgeable)
{
   PAP
      {
      1,       /* flow quantum */
      "",      /* AppleTalk address (filled in by Chooser) */
      0, 0, 0, 0
      };
};

resource 'comm' (-4095, sysHeap, purgeable)
{
   Serial
      {
      baud9600,      /* output baud rate */
      noParity,      /* output parity */
      oneStop,       /* output stop bits */
      data8,         /* output data size */
      0x00010000,    /* output handshaking */
      0x00000000,
      baud9600,      /* input baud rate */
      noParity,      /* input parity */
      oneStop,       /* input stop bits */
      data8,         /* input data bits */
      0,             /* input handshaking */
      0,
      1024,          /* input buffer size */
      ".AIn",        /* input driver name;filled in by Chooser */
      ".AOut"        /* output driver name; filled in by Chooser*/
      };
};
```

```
resource 'comm' (-4094, sysHeap, purgeable)
{
   PrinterShare
      {
      "",      /* AppleTalk address, filled in by Chooser */
      0
      };
};
```

You use the buffering resource to control the size of the buffers that your driver uses to communicate with the printing device. You also specify timeout values for your communications requests in this resource. Listing 3-23 shows the buffering resource that the ImageWriter II printer driver uses.

Listing 3-23 The buffering and input/output preferences resource for the ImageWriter II printer driver

```
resource gxUniversalIOPrefsType (gxUniversalIOPrefsID, sysHeap,
                                                      purgeable)
{
   standardIO,
   4,             /* 4 buffers */
   2048,          /* 2 KB each (enough for 1 scan line of data) */
   10,            /* up to 10 I/O operations pending */
   1200,          /* open/close timeout of 1200 clock ticks */
   1200           /* read/write timeout of 1200 clock ticks */
};
```

Defining Network Characteristics for Your Driver

You use capture (`'cpts'`) resources to define network strings for your driver if your printing device supports network capture and removal. These resources are only needed if you are using QuickDraw GX's default implementation of the Printer Access Protocol (PAP) communications interface.

If you are using the default PAP implementation, you define four capture strings, each of which is used by QuickDraw GX to manage the capture and removal of your printing device. The capture strings can contain special substitution strings, as described in the section “The Capture ('cpts') Resource” beginning on page 6-63 in the chapter “Printing Resources.”

Listing 3-24 shows the capture resources for the ImageWriter II printer driver.

Listing 3-24 Capture resources for the ImageWriter II printer driver

```
resource gxCaptureType (gxCapturedAppleTalkType, sysHeap,
                                                   purgeable)
{
   "\0D011ImageShared"
};

resource gxCaptureType (gxUncapturedAppleTalkType, sysHeap,
                                                  purgeable)
{
   "\0D011ImageWriter"
};

resource gxCaptureType (gxCaptureStringID, sysHeap, purgeable)
{
   "\0X1B\0X62NAMELENPRINTERNAMETYPELENPRINTERTYPE\0X01*"
};

resource gxCaptureType (gxReleaseStringID, sysHeap, purgeable)
{
   "\0X1B\0X62NAMELENPRINTERNAMETYPELENPRINTERTYPE\0X01*"
};
```

Defining Status Messages

You can define your own status (`'stat'`) resources for sending informational messages to the desktop printer window while a document is printing. Each status resource includes a status-type indicator, a status ID value, an alert value, and the status string. Listing 3-25 shows a status resource that defines the status text string displayed while the ImageWriter II printer driver is sending document data to the printer.

Listing 3-25 A status resource for the ImageWriter II printer driver

```
resource 'stat' (kDriverStatus, sysHeap, purgeable)
{
   'IWII',

   {
   gxInformationalStatus, 1, 0, "Sending data to printer";
   gxUserAlert, 1, kDriverStatus, "Please check that the printer
is on-line";
   }
};
```

Resources for Compatibility With the Macintosh Printing Manager

You can customize the way that your driver supports Macintosh Printing Manager printing by defining a customization (`'cust'`) resource. QuickDraw GX provides a default version, so this an optional resource.

The customization resource defines the resolution, translator settings for the driver, the pattern stretch factor, and how it translates Macintosh Printing Manager calls into QuickDraw GX printing messages. You can also fine tune the handling of certain geometries by the QuickDraw GX translator. The customization resource is described on page 6-47. The program that translates Macintosh Printing Manager calls to QuickDraw GX messages is described in *Inside Macintosh: QuickDraw GX Environment and Utilities*.

The customization resource from the ImageWriter II printer driver is shown in Listing 3-26.

Listing 3-26 The customization resource for the ImageWriter II printer driver

```
resource gxCustType (gxCustID, sysHeap, purgeable)
{
   144,144,                /* 300 dpi device */
   defaultUpDriver,        /* use LaserWriter interface */
   {2,2},                  /* pattern stretch of 2 */
   gxOptimizedTranslation  /* use default translator settings */
}
```

Defining Device Characteristics Specific to an Imaging System

Most of the unique qualities and characteristics of the printing device for which you are implementing a driver are described in the resources that you provide. The ImageWriter II printer driver includes three resources for the raster imaging system. You need to include resource definitions for the PostScript imaging system if you are developing a driver for a PostScript printing device. You need to include resource definitions for the vector imaging system if you are developing a driver for a vector printing device. All of these resources are described in the chapter “Printing Resources” in this book.

You must include a raster preferences (`'rdip'`) resource for a raster printer driver. This resource specifies imaging options for your driver, including color information. The raster preferences resource for the ImageWriter II printer driver, which is shown in Listing 3-27, includes color information for each color plane provided by the printing device.

Listing 3-27 The raster preferences resource for the ImageWriter II printer driver

```
resource gxRasterPrefsType (gxRasterPrefsID, sysHeap, purgeable)
{
   gxDefaultRaster,        /* default options are fine */

   0x00900000,0x00900000,  /* 144X144 dpi device */
   16,                     /* min band size == 2 head heights */
   0,                      /* max band size (0 is full page) */
   0x00004000,             /* RAM percentage (25%) */
   100*1024,               /* RAM slop (100K) */
   4,                      /* 4-bit device */
      {
      /* dithering offscreen */
      3,
      gxDontSetHalftone + gxDotTypeIsDitherLevel,
      0x002D0000,          /* angle unused for dithering */
      0x003C0000,          /* freq unused for dithering */
      4,                   /* dithering with level of 4 */
      gxLuminanceTint,     /* tint space unused for dithering */
                           /* dot color & background unused */
      gxRGBSpace, gxNoProfile, 0, 0, 0, 0,
      gxRGBSpace, gxNoProfile, 0xFFFF, 0xFFFF, 0xFFFF, 0,
      gxRGBSpace,          /* halftone space */
      gxIndexedSpace,      /* indexed color space */
      gxPrintingDriverBaseID, /* the color set to use */
      1 /* the color profile to use */
      };
};
```

The raster package (`'rpck'`) resource controls how bitmap data is packed into rasters for your driver. QuickDraw GX provides a default version of this resource, so providing your own version is optional. Listing 3-28 shows the raster package resource for the ImageWriter II driver.

Listing 3-28 The raster package resource for the ImageWriter II printer driver

```
resource gxRasterPackType (gxRasterPackID, sysHeap, purgeable)
{
   /*
      The packing buffer size. For the ImageWriter II, this
      is the # of bytes in the largest single packaged line.
   */
   2500,
   4,             /* this is CMYK (so colorsPasses == 4) */
   16,            /* print head is 16 pixels high */
   2,             /* it takes 2 passes to achieve the 16 pixels */
   1,             /* there is a 1 pixel difference between
                      these two passes */
   gxInterlaceColor,/* avoid ribbon contamination */
};
```

The final raster resource provided by the ImageWriter II driver is the raster package controls (`'ropt'`) resource, which you use to define how some forms of line feeding are performed on your printing device. Listing 3-29 shows the ImageWriter II printer driver version of this resource.

Listing 3-29 The raster package controls resource for the ImageWriter II printer driver

```
resource gxRasterPackOptionsType (gxRasterPackOptionsID, sysHeap,
                                                    purgeable)
{
   gxPrintingBaseID,
   gxPrintingBaseID + 10,

   /* forward line-feed characteristics */
   98,                  /* max line-feed amount is 98 */
   gxRasterNumToASCII,  /* express line-feed as ASCII */
   2,                   /* minimum width is 2 */
   "0",                 /* and pad with zeros */
   "\0X1BT",            /* <esc>T == set line-feed size */
   "\0X1Bf\0X0A",       /* <esc>f<lf> == direction forward,
                             do line feed */

   /* reverse line-feed characteristics */
   98,                  /* max line-feed amount is 98 */
   gxRasterNumToASCII,  /* express line-feed as ASCII */
   2,                   /* minimum width is 2 */
   "0",                 /* and pad with zeros */
   "\0X1BT",            /* <esc>T == set line-feed size */
   "\0X1Br\0X0A",       /* <esc>r<lf> == direction reverse,
                             do line feed */
};
```

User Interface and Chooser Support

You need to provide at least a minimal user interface to your printer driver, including support for the user to select printing choices, an interface to the Chooser for selecting your printer driver, and a collection of icons to represent your printer driver on the user's screen.

The contents of the the `ChooserSupport.r` file, which contains the Chooser resource definitions for the ImageWriter II driver, are shown in the QuickDraw GX sample code. The icon definitions for the ImageWriter II driver are found in the `newapp.r` resource file, which is also shown in the QuickDraw GX sample code.

Providing Printing Choices

You can use the dialog panel (`'ppnl'`) resources to provide QuickDraw GX with information to be displayed in dialog panels.

- The dialog panel resource gives QuickDraw GX the information necessary to display a panel.
- The extended dialog panel resource (resource type `'xdtl'`) provides QuickDraw GX with the information it needs to execute panel controls.

These resources are described in *Inside Macintosh: QuickDraw GX Printing*.

You can also modify the choices that are displayed to the user in the print dialog boxes by changing the values in the job collection. You can access and modify these values by using the appropriate tags (as described in the section "Using the Printing-Related Collections" beginning on page 3-20). Among the items that you might need to modify are the following:

- paper-tray availability
- print-quality choices
- print-scaling boundaries
- printing-file options

The Chooser and Your Driver

You need to define resources for your printer driver to provide a Chooser interface, which allows the user to select your driver from among the printer drivers available on the computer.

The Chooser uses the look (`'look'`) resource to generate the pop-up menu list of "Connect via:" options that it displays when the user selects your driver. Listing 3-30 shows an example of a look resource for the ImageWriter II printer driver, which supports serial, PAP, and PrinterShare connections.

Listing 3-30 The look resource for the ImageWriter II printer driver

```
resource 'look' (-4096, sysHeap, purgeable)
{
   2,    /* use the second in the list as the default */
   {
   "AppleTalk",   -4096,   isAppleTalk,   "ImageWriter";
   "Serial",      -4095,   iconCells,     "Modem Port";
   "Servers",     -4094,   isAppleTalk+isPrinterShare,
                           "ImageWriterIIIS";
   };
};
```

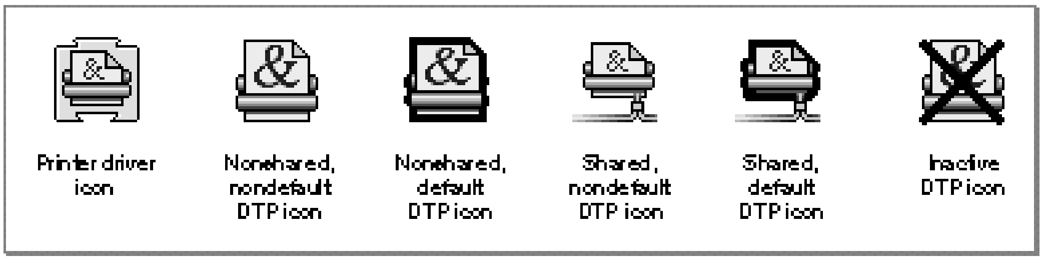
The look resource accesses each communications resource that you specify for your driver to determine parameter values for the type of connection that the user selects.

Defining Desktop Printer Icons for Your Printer Driver

You need to provide six icons for your printer driver: one to represent your driver in the Extensions folder inside of the System folder, and five others to represent different states of the desktop printer. Figure 3-3 shows the six icons for the Apple LaserWriter driver.

QuickDraw GX automatically imposes certain status icons over the desktop printer (DTP) icons to indicate certain conditions to the user. The status icons are described in the next section.

Figure 3-3 The Apple LaserWriter printer driver icons


Printer driver icon
Nonshared, nondefault DTP icon
Nonshared, default DTP icon
Shared, nondefault DTP icon
Shared, default DTP icon
Inactive DTP icon

QuickDraw GX uses the printer driver icon to represent the driver in the Extensions folder in the System folder. The desktop printer icons are used to represent different states of the driver when the user has made it a desktop printer.

- The nonshared, nondefault desktop printer icon is displayed on the desktop when the printer is not the default printer and is not a shared printer. This icon is also used to represent your printer driver in the Chooser.
- The nonshared, default desktop printer icon is displayed on the desktop when the user has selected a nonshared printer as the default printer.
- The shared, nondefault desktop printer icon is displayed on the desktop of both the client and server computers when a shared printer is not the default printer.
- The shared, default desktop printer icon is displayed on the desktop of both the client and server computers when a user has selected a shared printer as the default printer.
- The inactive desktop printer icon is displayed when a desktop printer is not on the startup disk desktop or when QuickDraw GX is not active (for example, if the user has installed QuickDraw GX but has booted the computer with extensions turned off).

You need to follow certain guidelines when designing your icons:

- Each default desktop printer icon must resemble the icon for the corresponding nondefault desktop printer icon and must have a bold (3 pixels wide) outline drawn around it.
- Each shared desktop printer icon must resemble the printing device with networking cables slightly overlapping at the bottom of the printing device.
- The inactive desktop printer icon must resemble the nonshared, nondefault desktop printer icon and must have bold (3 pixels wide) crossing lines drawn over it.

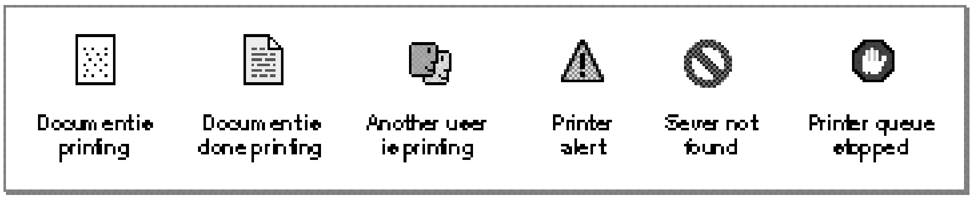
One strategy for designing your icons is to design your nondefault desktop printer icons with room for the border that is added for their “default” versions. This simplifies your design process and provides the user with icons that have a consistent appearance and are not distorted.

You need to carefully place the network cables in your shared desktop printer icons because of how they appear when the status icons are overlaid on them. The status icons are more noticeable when the cables are up one pixel from the edge and when the “Y” area of the cables are located in the right half of the icon space. The status icons are described in the next section.

The Desktop Printer Status Icons

QuickDraw GX uses the desktop-printer status icons to convey additional printing information to the user. These are stand-alone icons that are imposed on the lower-left quadrant of the desktop printer icons whenever the status of a desktop printer changes. Each of the status icons is designed with unique color characteristics to help visually distinguish them. Figure 3-4 shows the QuickDraw GX desktop-printer status icons.

Figure 3-4 The QuickDraw GX desktop-printer status icons

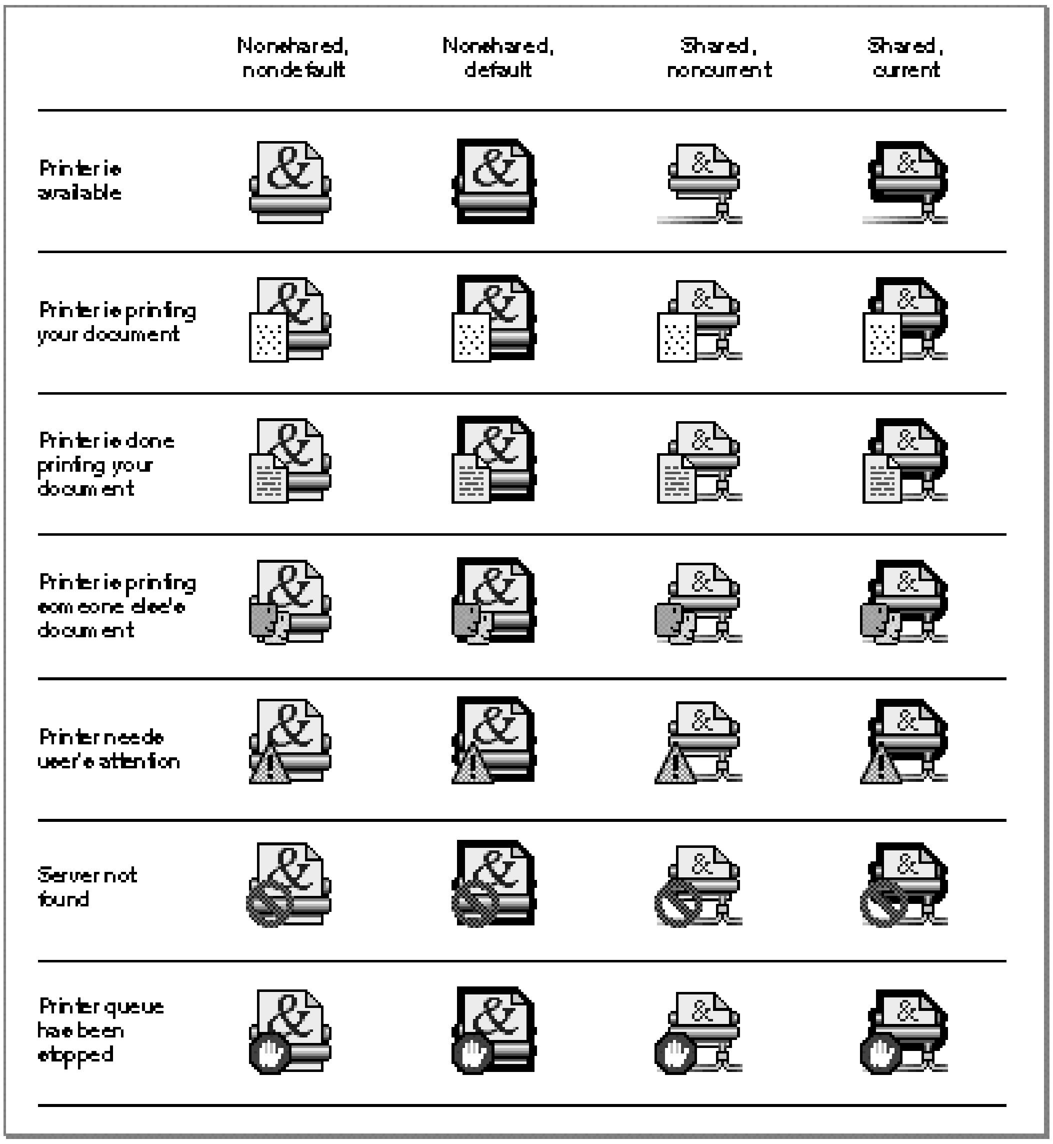


When a desktop printer is available but not currently being used, the desktop printer icon is displayed without any status added. QuickDraw GX automatically adds the status icon whenever the printer’s status changes, as follows:

- When one of the user’s documents is printing on the desktop printer, the “document is printing” status icon is overlaid on the desktop printer icon.
- When one of the user’s documents has finished printing on the desktop printer, the “document is done printing” status icon is overlaid on the desktop printer icon.
- When a shared desktop printer is printing a document that was sent to it by another user, the “another user is printing” status icon is overlaid on the desktop printer icon.
- When a desktop printer requires user attention for conditions such as paper jams or an empty paper tray, the “printer alert” status icon is overlaid on the desktop printer icon.
- When the desktop printer server cannot be found on the network, the “server not found” status icon is overlaid on the desktop printer icon.
- When the user chooses the Stop Printer Queue item in the Printing menu, the “printer queue stopped” status icon is overlaid on the desktop printer icon.

You provide four desktop printer icons for your printer driver, and QuickDraw GX automatically overlays six different icons on them to convey status information. This means that a desktop printer can be represented on the user’s desktop by 28 different icons, as shown in Figure 3-5.

Figure 3-5 Desktop printer icons showing printer status


Nonshared, nondefault
Nonshared, default
Shared, noncurrent
Shared, current
Printer is available
Printer is printing your document
Printer is done printing your document
Printer is printing someone else's document
Printer needs user's attention
Server not found
Printer queue has been stopped

Bundling Your Printer Driver Icons

You need to create a bundle (`'BNDL'`) resource for your printer driver, just as you do for any Macintosh application program. The bundle resource, which is described in *Inside Macintosh: Macintosh Toolbox Essentials*, associates your printer driver with its icons and with any files that it creates.

QuickDraw GX needs to map the local IDs of your desktop printer icons (which are described in the previous section) into specific IDs to properly use them. For this to happen, you must define a file reference (`'FREF'`) resource for each type of desktop printer icon that you include with your driver. The file types listed in Table 3-11 must be used as shown.

Table 3-11 File types for desktop printer icons

Icon file type	Icon usage
`'dpnn'`	Nonshared, nondefault desktop printer
`'dpcn'`	Nonshared, default desktop printer
`'dpns'`	Shared, nondefault desktop printer
`'dpcs'`	Shared, default desktop printer
`'dvcl'`	Desktop printer when QuickDraw GX is not active
`'dppz'`	Printer driver in the Extensions folder when QuickDraw GX is active
`'pdvr'`	Printer driver in the Chooser when QuickDraw GX is active, and printer driver in the Extensions folder when QuickDraw GX is not active

The bundle resource for the ImageWriter II printer driver defines the local ID and file reference IDs for each of the desktop printer icon definitions that follow. The resource definition is shown in Listing 3-31.

Listing 3-31 The bundle resource for the ImageWriter II printer driver

```
resource 'BNDL' (gxPrintingDriverBaseID + 1, sysHeap, purgeable)
{
   kCreatorType,
   0,
   {
      'ICN#', { 0, gxPrintingDriverBaseID + 2;
                1, gxPrintingDriverBaseID + 3;
                2, gxPrintingDriverBaseID + 4;
                3, gxPrintingDriverBaseID + 5
               },
      'FREF', { 0, gxPrintingDriverBaseID + 2;
                1, gxPrintingDriverBaseID + 3;
                2, gxPrintingDriverBaseID + 4;
                3, gxPrintingDriverBaseID + 5;
                0, gxPrintingDriverBaseID + 1
               }
   }
};
```

The ImageWriter II printer driver includes a file reference resource for its file type signature and one file reference resource for each type of desktop printer icon, as shown in Listing 3-32.

Listing 3-32 The file reference resources for the ImageWriter II printer driver

```
resource 'FREF' (gxPrintingDriverBaseID + 1, sysHeap, purgeable) {
kFileType, 0, "" };

resource 'FREF' (gxPrintingDriverBaseID + 2, sysHeap, purgeable) {
'dpnn', 0, "" };

resource 'FREF' (gxPrintingDriverBaseID + 3, sysHeap, purgeable) {
'dpns', 1, "" };

resource 'FREF' (gxPrintingDriverBaseID + 4, sysHeap, purgeable) {
'dpcn', 2, "" };

resource 'FREF' (gxPrintingDriverBaseID + 5, sysHeap, purgeable) {
'dpcs', 3, "" };
```

The ImageWriter II printer driver defines icons in various sizes and resolution for display on the user's desktop. Note that each related icon (the various sizes and resolutions for each purpose) shares the same resource ID. For example, each of the icons used to represent a desktop printer that is shared and default (`'dpcs'`) has resource ID `gxPrintingDriverBaseID + 5`, as shown in Listing 3-33. The actual data for the icons can be found in the QuickDraw GX sample code.

Listing 3-33 The icon resources for the ImageWriter II printer driver

```
/*
   Icons in various sizes and resolutions for the different
    representations of the desktop printer are included here.
*/

   /* nonshared, nondefault desktop printer icon definitions*/
resource 'ics#' (gxPrintingDriverBaseID + 2, sysHeap, purgeable)
{        /* icon resource goes here */
};

resource 'ics4' (gxPrintingDriverBaseID + 2, sysHeap, purgeable)
{        /* icon resource goes here */
};
```

```
resource 'ics8' (gxPrintingDriverBaseID + 2, sysHeap, purgeable)
{        /* icon resource goes here */
};

resource 'ICN#' (gxPrintingDriverBaseID + 2, sysHeap, purgeable)
{        /* icon resource goes here */
};

resource 'icl4' (gxPrintingDriverBaseID + 2, sysHeap, purgeable)
{        /* icon resource goes here */
};

resource 'icl8' (gxPrintingDriverBaseID + 2, sysHeap, purgeable)
{        /* icon resource goes here */
};

   /* nondefault, shared desktop printer icon definitions*/
resource 'ICN#' (gxPrintingDriverBaseID + 3, sysHeap, purgeable)
{        /* icon resource goes here */
};

resource 'icl4' (gxPrintingDriverBaseID + 3, sysHeap, purgeable)
{        /* icon resource goes here */
};

resource 'icl8' (gxPrintingDriverBaseID + 3, sysHeap, purgeable)
{        /* icon resource goes here */
};

   /* default, nonshared desktop printer icons */
resource 'ICN#' (gxPrintingDriverBaseID + 4, sysHeap, purgeable)
{        /* icon resource goes here */
};

resource 'icl4' (gxPrintingDriverBaseID + 4, sysHeap, purgeable)
{        /* icon resource goes here */
};

resource 'icl8' (gxPrintingDriverBaseID + 4, sysHeap, purgeable)
{        /* icon resource goes here */
};

   /* default, shared desktop printer icon definitions*/
resource 'ICN#' (gxPrintingDriverBaseID + 5, sysHeap, purgeable)
{        /* icon resource goes here */
};
```

```
resource 'icl4' (gxPrintingDriverBaseID + 5, sysHeap, purgeable)
{        /* icon resource goes here */
};

resource 'icl8' (gxPrintingDriverBaseID + 5, sysHeap, purgeable)
{        /* icon resource goes here */
};
```

CHAPTER 4

Printing Messages

Contents

QuickDraw GX implements printing as a series of messages that it sends to printing extensions, printer drivers, and applications. You need to override some of the QuickDraw GX printing messages to develop either a printing extension or a printer driver. You choose which messages to override based on what functionality your driver or extension is providing.

This chapter describes the messages you can override in either printing extensions or printer drivers. Application programs can also override many of these messages. The way in which you use these messages to develop printing extensions is described in the chapter “Printing Extensions” in this book; the way in which you use these messages to develop printer drivers is described in the chapter “Printer Drivers” in this book.

Before reading this chapter you need to understand how the Message Manager handles messages, including how to send and forward messages. The Message Manager is described in *Inside Macintosh: QuickDraw GX Environment and Utilities.* You also need to understand the QuickDraw GX printing architecture, which is described in *Inside Macintosh: QuickDraw GX Printing.*

You need to read this chapter if you are developing a printing extension or a printer driver for use with QuickDraw GX. This chapter constitutes the complete reference guide to the messages that you can use to develop these programs.

This chapter begins with a brief overview of how QuickDraw GX uses message passing to implement printing and then describes

- the terminology used to describe printing messages
- the categories of printing messages and which messages fit into each category
- the data structures that are used by the printing messages
- each of the messages that you can use to develop printing extensions and printer drivers

About the Printing Messages

The printing messages described in this chapter are sent by QuickDraw GX during the process of printing a document. QuickDraw GX provides default implementations of many of these messages that perform the basic task for which the message is intended. This means that you only need to override those messages that apply to the specifics of your extension or driver. And many of the specifics of a device are defined in the resources that are included in your extension or driver.

For example, since QuickDraw GX provides robust default implementations, you can create a printer driver for a PostScript printer by defining resources and overriding only one message, the `GXSetupImageData` message. You need to override many messages for some printers and only a few messages for others; however, in most cases the default implementation supplied by QuickDraw GX provides functionality that you want to use. You can override messages to add to or modify that functionality as you need to for your task or device.

Printing Message Terminology

This chapter uses object-oriented programming terminology to describe how the printing messages operate. This section reviews the terms used in this chapter to describe the messages. For a more complete description of these terms and how message-passing works in QuickDraw GX, read the chapter “Message Manager” in *Inside Macintosh: QuickDraw GX Environment and Utilities.* In this book, the chapter “Introduction to Printing Extensions and Drivers” contains an overview of how message-passing works with printing extensions and printer drivers.

The QuickDraw GX printing architecture uses a number of different object types to provide printing. Each print object is an abstract data type that encapsulates certain properties that you can use QuickDraw GX functions to access. The objects used in the QuickDraw GX printing architecture are described in *Inside Macintosh: QuickDraw GX Printing*. These objects include job objects, printer objects, format objects, and paper-type objects.

When QuickDraw GX needs to accomplish a printing task, it sends a message. For example, QuickDraw GX sends a message when it needs a driver to establish a communications connection with a printing device, when it wants to print a page, and when it needs a printer driver to convert a bitmap into a data format for a raster printer.

You respond to messages by overriding them. Your override of a message intercepts the message and takes some action. A number of message handlers can respond to each message, including the system software, the application program, a printer driver, and one or more printing extensions.

QuickDraw GX provides a default implementation of each printing messages. You can partially override a message to add to the default implementation’s response or to change the results of the default implementation. Depending on the nature of a message, you can forward it to the other handlers and then perform your tasks, or you can perform your tasks and then forward it to other handlers.

Note
The action of the default implementation of each printing message is noted in this chapter. Some of the default implementations provided by QuickDraw GX are empty (all that the function does is return). u

You can totally override some messages, which means that your override completely replaces the default implementation and does not foward the message. The “Special Considerations” section for each of the messages described in the section “Printing Messages Reference” in this chapter indicates which printing messages you can totally override. If you do totally override a printing message that requires forwarding, the default implementation is not invoked, which means that a vital operation might be neglected and serious errors might result.

Using the Printing Messages

QuickDraw GX sends printing messages to message handlers in the message chain, including the application program, any printing extensions that are active, and the printer driver. You can override any of these messages in your printing extensions and printer drivers to modify or replace the default implementation of the message that is provided by QuickDraw GX. You can also send some of the printing messages yourself to accomplish certain tasks.

You can override printing messages to

- manage the job context
- manipulate print objects
- modify application-initiated actions
- write data to spool files
- retrieve data from spool files
- interact with users through dialog boxes
- convert spool files to universal images
- send raster image data
- send PostScript data
- send vector image data
- communicate with devices
- convert a print record in order to use QuickDraw GX to print documents originally created with QuickDraw
- display printing status messages
- add menu items to the Finder's Printing menu

The chapters "Printing Extensions" and "Printer Drivers" in this book describe how you can use the printing messages to develop your extensions and drivers. This chapter provides the reference material that you need to understand and use the messages properly.

Printing Messages Reference

This section describes data types and messages that are used to develop QuickDraw GX printing extensions and printer drivers. The resources that you use to develop extensions and drivers are described in the chapter "Printing Resources" in this book.

Various sections show the structures and enumerations that are used with the printing messages.

The “Printing Messages” section describes the interface and functionality of each message that you can override in a printing extension or printer driver.

A number of the printing messages take parameters that are data structures. This section describes those data structures and the constants (enumerated types) that are used to define values within them.

The constants and data types are grouped according to usage, as follows:

- printer driver constants and data types
- Macintosh Printing Manager compatability constants and data types
- raster driver constants and data types
- PostScript driver constants and data types
- vector driver constants and data types
- user interface constants and data types

Data Types for Printer Drivers

This section describes the data types that are used by printer drivers.

The Print-to-File Structure

The print-to-file structure, of data type `gxPrintDestinationRec`, is used to specify information about writing a document to file. This structure defines the parameters that specify how the document is stored in a file.

```
struct gxPrintDestinationRec {
   Boolean     printToFile;
   FSSpec      fSpec;
   Boolean     includeFonts;
   Str31       fileFormat;
};

typedef struct gxPrintDestinationRec gxPrintDestinationRec,
*gxPrintDestinationPtr, **gxPrintDestinationHdl;
```

Field descriptions

`printToFile`	A Boolean value that is `true` if the current document is being printed to a file and `false` if not.
`fSpec`	The `FSSpec` structure for the file.
`includeFonts`	A Boolean value that is `true` if system fonts are included in the file and `false` if not. Application fonts are always included in the file.
`fileFormat`	The name of the file format to use for writing the file.

The Printing Buffer Structure

The printing buffer structure, of data type `gxPrintingBuffer`, holds the document data that QuickDraw GX sends to the printer. It is used with the `GXDumpBuffer` and `GXFreeBuffer` messages, which are described in the section "Device Communications Messages" beginning on page 4-131.

```
struct gxPrintingBuffer {
   long     size;
   long     userData;
   char     data[1];
};

typedef struct gxPrintingBuffer gxPrintingBuffer;
```

Field descriptions

`size`	The number of bytes in the buffer.
`userData`	The signature for this buffer, into which a message handler can write a unique ID. If you are writing a driver and overriding the `GXDumpBuffer` and `GXFreeBuffer` messages, you can use this value to match your buffers.
`data`	The data in the buffer. This is an array containing `size` bytes.

The Page Information Structure

The page information structure, of data type `gxPageInfoRecord`, specifies information about a page that is being printed. It is used with the `GXRenderPage` message, which is described on page 4-96.

```
struct gxPageInfoRecord {
   long     docPageNum;
   long     copyNum;
   Boolean  formatChanged;
   Boolean  pageChanged;
   long     internalUse;
};

typedef struct gxPageInfoRecord gxPageInfoRecord;
```

Field descriptions

`docPageNum`	The number of the page about which this structure contains information.
`copyNum`	The copy number of the page that is being printed.
`formatChanged`	A Boolean value that is `true` if this page uses a different format than the previous (most recently printed) page and `false` if not.

`pageChanged`	A Boolean value that is `true` if this page has different contents than the previous (most recently printed) page and `false` if not.
`internalUse`	A private field for internal use.

Constants and Data Types for Macintosh Printing Manager Compatibility

This section describes the structure and enumerations that are used to define the universal print structure. This structure allows applications that were designed to print with the Macintosh Printing Manager to print with QuickDraw GX.

The Universal Print Structure

The universal print structure, of data type `gxUniversalPrintRecord`, contains information about the current print job. You need to know about the format of this structure if you are converting information from a print record used with the Macintosh Printing Manager. This structure is used instead of the Macintosh Printing Manager `TPrint` structure. It is used with the `GXConvertPrintRecordFrom`, `GXConvertPrintRecordTo`, and `GXPrintRecordToJob` messages, which are described in the section “Compatibility Messages” beginning on page 4-147. The print record used with the Macintosh Printing Manager is described in *Inside Macintosh: Imaging With QuickDraw*.

```
struct gxUniversalPrintRecord {
   short    prVersion;

   short    appDev;
   short    appVRes;
   short    appHRes;
   Rect     appPage;
   Rect     appPaper;

   short    devType;
   short    pageV;
   short    pageH;
   char     filler;
   char     feed;

   short       devKind;
   short       devVRes;
   short       devHRes;
   Rect        devPage;

   short       actualCopies;
   short       options;
   short       reduction;
   char        orientation;

   char        qualityMode;
   char        firstTray;
   char        remainingTray;

   char        coverPage;
   char        headMotion;
   char        saveFile;

   char        userCluster1;
   char        userCluster2;
   char        userCluster3;

   short       firstPage;
   short       lastPage;
   short       copies;
   char        reserved1;
   char        reserved2;
   PrIdleProcPtr  pIdleProc;
   Ptr         pFileName;
   short       fileVol;
   char        fileVers;
   char        reserved3;

   short    printX[19];
};

typedef struct gxUniversalPrintRecord gxUniversalPrintRecord,
*gxUniversalPrintRecordPtr, **gxUniversalPrintRecordHdl;
```

Field descriptions

`prVersion`	The version ID of this converted print record. This value is always the constant `gxPrintRecordVersion`, a value of 8.
`appDev`	The kind of device this print record applies to. At this time, this field always has the value 0.
`appVRes`	The vertical resolution of the application.
`appHres`	The horizontal resolution of the application.
`appPage`	The page size for this print record, stated in terms of the application resolution.
`appPaper`	The paper rectangle for this print record. This value is an offset from the value of the `appPage` field (the page always fits within the paper, and the edge of the paper is a negative offset from the page).
`devType`	The device type for this print record. At this time, the value of this field is always 0.

`pageV`	The vertical page height in 120ths of an inch.
`pageH`	The horizontal page width in 120ths of an inch.
`filler`	Unused.
`feed`	The feed mode used for this print record. The possible values are given in the section “Feed Mode Options for the Universal Print Structure” on page 4-15.
`devKind`	The kind of device for this print record. At this time, the value of this field is always `0xA900`.
`devVRes`	The vertical resolution of the device.
`devHRes`	The horizontal resolution of the device.
`devPage`	The page size of the device.
`actualCopies`	The actual number of copies for this print job.
`options`	The options for this device. The constants that you can combine into a single value for this field are given in the “Print Options for the Universal Print Structure” on page 4-15.
`reduction`	The reduction or enlargement factor used for this print job.
`orientation`	The orientation of the paper. The possible values are given in the section “Paper Orientation Options for the Universal Print Structure” on page 4-16.
`qualityMode`	The quality mode used for this print job. The possible values are given in the section “Print Quality Modes for the Universal Print Structure” on page 4-17.
`firstTray`	Which of the printer’s paper trays is used as the first paper tray for this print job. The possible values are given in the section “Paper Tray Selections for the Universal Print Structure” on page 4-17.
`remainingTray`	The other paper tray used for this print job. The possible values are given in the section “Paper Tray Selections for the Universal Print Structure” on page 4-17.
`coverPage`	The type of cover page to print for this print job. The possible values are given in the section “Cover Page Options for the Universal Print Structure” on page 4-18.
`headMotion`	The type of print-head motion to use for this print job. The possible values are given in the section “Print-Head Motions for the Universal Print Structure” on page 4-18.
`saveFile`	The type of file to save for this print job. The possible values are given in the section “File Save Types for the Universal Print Structure” on page 4-19. This field applies only to PostScript print jobs.
`userCluster1`	Unused.
`userCluster2`	Unused.
`userCluster3`	Unused.
`firstPage`	The number of the first page.
`lastPage`	The number of the last page.
`copies`	The number of copies. This value is always 1.
`reserved1`	Unused.

`reserved2`	Unused.
`pIdleProc`	A pointer to the idle procedure to use with this print job.
`pFileName`	A pointer to the name of the spool file for this print job.
`fileVol`	The volume reference number of the spool file volume.
`fileVers`	The spool file version, which must be 0.
`reserved3`	Reserved. This field must have a value of 0.
`printX`	Reserved for internal use.

Feed Mode Options for the Universal Print Structure

You can use the feed mode constants to define the value of the `feed` field in the universal print structure.

```
enum {
   gxAutoFeed     = 0,
   gxManualFeed   = 1
};
```

Constant descriptions

`gxAutoFeed`	The print job uses automatic paper feeding.
`gxManualFeed`	The print job uses manual paper feeding.

Print Options for the Universal Print Structure

You can use the print option constants to define the value of the `options` field in the universal print structure. You can combine these constants into a single value.

```
enum {
   gxPreciseBitmap      = 0x0001,
   gxBiggerPages        = 0x0002,
   gxGraphicSmoothing   = 0x0004,
   gxTextSmoothing      = 0x0008,
   gxFontSubstitution   = 0x0010,
   gxInvertPage         = 0x0020,
   gxFlipPageHoriz      = 0x0040,
   gxFlipPageVert       = 0x0080,
   gxColorMode          = 0x0100,
   gxBidirectional      = 0x0200,
   gxUserFlag0          = 0x0400,
   gxUserFlag1          = 0x0800,
   gxUserFlag2          = 0x1000,
   gxReservedFlag0      = 0x2000,
   gxReservedFlag1      = 0x4000,
   gxReservedFlag2      = 0x8000
};
```

Constant descriptions

`gxPreciseBitmap`	The driver needs to format pages as tall-adjusted for the Apple ImageWriter family of printers and to use precise bitmaps for the Apple LaserWriter family of printers.
`gxBiggerPages`	The driver needs to not apply gaps if printing this job to one of the Apple ImageWriter family of printers, and the driver needs to use a large print area for the Apple LaserWriter family of printers.
`gxGraphicSmoothing`	The driver needs to perform graphics smoothing on the Apple LaserWriter family of printers.
`gxTextSmoothing`	The driver needs to perform text smoothing for this print job.
`gxFontSubstitution`	The driver needs to perform font substitution for this print job.
`gxInvertPage`	The driver needs to invert the printed image (convert white to black and black to white) for this print job.
`gxFlipPageHoriz`	The driver needs to flip pages horizontally for this print job.
`gxFlipPageVert`	The driver needs to flip pages vertically for this print job.
`gxColorMode`	This print job uses color printing.
`gxBidirectional`	This print job uses bidirectional printing.

Paper Orientation Options for the Universal Print Structure

You can use the paper orientation option constants to define the value of the `orientation` field in the universal print structure.

```
enum {
   gxPortraitOrientation      = 0,
   gxLandscapeOrientation     = 1,
   gxAltPortraitOrientation   = 2,
   gxAltLandscapeOrientation  = 3
};
```

Constant descriptions

`gxPortraitOrientation`	The driver needs to print the print job in portrait orientation.
`gxLandscapeOrientation`	The driver needs to print the print job in landscape orientation.

`gxAltPortraitOrientation`	The driver needs to print the print job in rotated (90 degrees) portrait orientation.
`gxAltLandscapeOrientation`	The driver needs to print the print job in rotated (90 degrees) landscape orientation.

Print Quality Modes for the Universal Print Structure

You can use the print quality mode constants to define the value of the `qualityMode` field in the universal print structure.

```
enum {
   gxBestQuality             = 0,
   gxFasterQuality           = 1,
   gxDraftQuality            = 2
};
```

Constant descriptions

`gxBestQuality`	The driver needs to print this print job with highest-quality printing.
`gxFasterQuality`	The driver needs to print this print job with medium-quality printing.
`gxDraftQuality`	The driver needs to print this print job with fastest-quality printing.

Paper Tray Selections for the Universal Print Structure

You can use the paper tray selection constants to define the value of the `firstTray` and `remainingTray` fields in the universal print structure.

```
enum {
   gxFirstTray    = 0,
   gxSecondTray   = 1,
   gxThirdTray    = 2
};
```

Constant descriptions

`gxFirstTray`	The printer’s first paper tray is used for this print job.
`gxSecondTray`	The printer’s second paper tray is used for this print job.
`gxThirdTray`	The printer’s third paper tray is used for this print job.

Cover Page Options for the Universal Print Structure

You can use the cover page option constants to define the value of the `coverPage` field in the universal print structure.

```
enum {
   gxNoCoverPage     = 0,
   gxFirstPageCover  = 1,
   gxLastPageCover   = 2
};
```

Constant descriptions

`gxNoCoverPage`	The driver need not print a cover page for this print job.
`gxFirstPageCover`	The driver needs to print a cover page before the first page for this print job.
`gxLastPageCover`	The driver needs to print a cover page after the last page for this print job.

Print-Head Motions for the Universal Print Structure

You can use the print-head motion constants to define the value of the `headMotion` field in the universal print structure.

```
enum {
   gxUnidirectionalMotion  = 0,
   gxBidirectionalMotion   = 1
};
```

Constant descriptions

`gxUnidirectionalMotion`	The driver needs to move the print head in only one direction for this print job.
`gxBidirectionalMotion`	The driver needs to move the print head in both directions for this print job.

File Save Types for the Universal Print Structure

You can use the constants that define how to save a print file to define the value of the `saveFile` field in the universal print structure.

```
enum {
   gxNoFile          = 0,
   gxPostScriptFile  = 1
};
```

Constant descriptions

`gxNoFile`	The driver need not save the print job in a file.
`gxPostScriptFile`	The driver needs to save the print job in a PostScript file.

Constants and Data Types for the Raster Imaging System

This section describes the enumerations and structures that are used by the raster imaging system messages. You need to know about these data types if you are writing a printing extension or printer driver for a printing device that uses the raster imaging system.

Raster Offscreen Structure

The raster offscreen structure, of type `gxOffscreenRec`, contains an offscreen area for a bitmap.

```
struct gxOffscreenRec {
   short        numberOfPlanes;
   Handle       offscreenStorage;
   gxOffscreenPlaneRec
                thePlanes[1];
};

typedef struct gxOffscreenRec gxOffscreenRec, *gxOffscreenPtr,
**gxOffscreenHdl;
```

Field descriptions

`numberOfPlanes`	The number of planes into which to draw the bitmap.
`offScreenStorage`	A handle to the bitmap’s image data.
`thePlanes`	An array of plane structures into which to draw the data. Each element of this array is an offscreen plane structure, which is described in the next section.

Raster Offscreen Plane Structure

The raster offscreen plane structure, of data type `gxOffscreenPlaneRec`, defines the format of one plane in the offscreen planar area.

```
struct gxOffscreenPlaneRec {
   gxViewPort     theViewPort;
   gxViewDevice   theDevice;
   gxViewGroup    theViewGroup;
   gxShape        theBitmap;
   gxBitMap       theBits;
};

typedef struct gxOffscreenPlaneRec gxOffscreenPlaneRec;
```

Field descriptions

`theViewPort`	The view port for this offscreen plane.
`theDevice`	The view device for this offscreen plane.
`theViewGroup`	The view group for this offscreen plane.
`theBitmap`	The shape object that represents the offscreen bitmap.
`theBits`	The actual bitmap data for this area.

Raster Offscreen Setup Structure

The raster offscreen setup structure, of data type `gxOffscreenSetupRec`, defines how to store raster data in offscreen memory.

```
struct gxOffscreenSetupRec {
   short       width;
   short       minHeight;
   short       maxHeight;
   Fixed       ramPercentage;
   long        ramSlop;
   short       depth;
   gxMapping   vpMapping;
   gxMapping   vdMapping;
   short       planes;
   gxPlaneSetupRec
               planeSetup[4];
};

typedef struct gxOffscreenSetupRec gxOffscreenSetupRec;
```

Field descriptions

`width`	The width in pixels of the raster.
`minHeight`	The minimum height in pixels. The actual height that was created for the offscreen storage is stored into this value.
`maxHeight`	The maximum height in pixels.
`ramPercentage`	The maximum percentage of available RAM to use for this storage.
`ramSlop`	The amount of RAM that must be left available after allocating for this storage.
`depth`	The depth, in bits, of each plane.
`vpMapping`	The mapping for offscreen view ports.
`vdMapping`	The mapping for offscreen view devices.
`planes`	The number of planes to allocate, each of which will have `depth` bits allocated for it.
`planeSetup`	The parameters of each plane, defined in the `gxPlaneSetupRec` structure, which is described in the next section. Four of these records are allocated because most drivers use four planes.

Raster Offscreen Plane Setup Structure

The raster offscreen plane setup structure, of data type `gxPlaneSetupRec`, defines settings for storing a single plane of raster data in offscreen memory. The raster offscreen setup structure, described in the previous section, uses four of these structures.

```
struct gxPlaneSetupRec {
   gxRasterPlaneOptions    planeOptions;
   gxHalftone              planeHalftone;
   gxColorSpace            planeSpace;
   gxColorSet              planeSet;
   gxColorProfile          planeProfile;
};

typedef struct gxPlaneSetupRec gxPlaneSetupRec;
```

Field descriptions

`planeOptions`	The options for this plane. This value is the combined values of the constants that you include from the raster plane options enumeration, which is described in the next section.
`planeHalftone`	The halftone structure for this plane. This is optional.
`planeSpace`	The color space for this plane. The color-space value constants are shown in the chapter "Colors and Color-Related Objects" in *Inside Macintosh: QuickDraw GX Objects*. You can specify the value `gxNoSpace` to use the default color space.

`planeSet`	The color set for this plane. You can specify `nil` to use the default color set.
`planeProfile`	The color profile for this plane. You can specify `nil` if you do not want color matching applied to this plane.

Raster Plane Options

The raster plane options enumeration defines constants that you can use in the `planeOptions` field of the raster offscreen plane setup structure.

```
enum {
   gxDefaultOffscreen      = 0x00000000,
   gxDontSetHalftone       = 0x00000001,
   gxDotTypeIsDitherLevel  = 0x00000002
};

typedef long gxRasterPlaneOptions;
```

Constant descriptions

`gxDefaultOffscreen`	The driver allocates bits for the print job and creates halftone values. This is the default value.
`gxDontSetHalftone`	The driver does not call the `GXSetViewPortHalftone` function.
`gxDotTypeIsDitherLevel`	The driver calls the `GXSetViewPortDither` function and uses the `gxDotType` constant as the dithering level.

Raster Package Bitmap Structure

The raster package bitmap structure, of data type `gxRasterPackageBitmapRec`, is used with the `GXRasterPackageBitmap` message.

```
struct gxRasterPackageBitmapRec {
   gxBitmap       *bitmapToPackage;
   unsigned short startRaster;
   unsigned short colorBand;
   Boolean        isBandDirty;
   Rect           dirtyRect;
};

typedef struct gxRasterPackageBitmapRec gxRasterPackageBitmapRec;
```

Field descriptions

`bitmapToPackage`	A pointer to the bitmap that contains the data to be packaged.
`startRaster`	The raster where the packaging is to begin.
`colorBand`	Which color pass of packaging this structure represents.
`isBandDirty`	A Boolean value that is `true` if the raster band being packaged contains any nonwhite bits and is `false` if the raster band is all white.
`dirtyRect`	A rectangle that defines the area in the bitmap that contains dirty bits.

Raster Imaging System Structure

The raster imaging system structure, of data type `gxRasterImageDataRec`, is the structure that QuickDraw GX uses to maintain information about the state of the raster imaging system. This structure is used with the `GXRasterLineFeed` and `GXRasterPackageBitmap` messages, which are described in the section "Raster Imaging Messages" beginning on page 4-97.

```
struct gxRasterImageDataRec {
   gxRasterRenderOptions      renderOptions;
   Fixed                      hImageRes;
   Fixed                      vImageRes;
   short                      minBandSize;
   short                      maxBandSize;
   gxRectangle                pageSize;
   short                      currentYPos;
   gxRasterPackageRec         packagingInfo;
   Boolean                    optionsValid;
   gxRasterPackageControlsRec packageControls;
   gxOffscreenSetupRec        theSetup;
};

typedef struct gxRasterImageDataRec gxRasterImageDataRec,
*gxRasterImageDataPtr, **gxRasterImageDataHdl;
```

Field descriptions

`renderOptions`	Rendering options for raster imaging. This value is the combined values of the constants that you include from the raster render options enumeration, which is described in the next section.
`hImageRes`	The horizontal resolution for imaging.
`vImageRes`	The vertical resolution for imaging.
`minBandSize`	The minimum band size to use, in pixels.
`maxBandSize`	The maximum band size to use, in pixels.
`pageSize`	The size of the page in pixels.

`currentYPos`	The current position on the page.
`packagingInfo`	The raster package structure.
`optionsValid`	A Boolean value that is `true` if options are specified for the packaging messages and `false` if not.
`packageControls`	The raster package controls structure.
`theSetup`	The setup for the offscreen planar area data. This is a variable length object.

Raster Render Options

The raster render options enumeration defines constants that you can use in the `renderOptions` field of the raster imaging system structure, which is described in the previous section.

```
enum {
   gxDefaultRaster                = 0x00000000,
   gxDontResolveTransferModes     = 0x00000001,
   gxRenderInReverse              = 0x00000002,
   gxOnePlaneAtATime              = 0x00000004,
   gxSendAllBands                 = 0x00000008
};

typedef long gxRasterRenderOptions;
```

Constant descriptions

`gxDefaultRaster`	The driver uses the default raster options.
`gxDontResolveTransferModes`	If this is included, the driver does not need to resolve transfer modes. The default is to resolve transfer modes.
`gxRenderInReverse`	The driver needs to traverse the raster image in reverse order.
`gxOnePlaneAtATime`	The driver needs to render each plane separately.
`gxSendAllBands`	The driver needs to send every band of data, even if it is all empty.

Raster Package Structure

The raster package structure, of data type `gxRasterPackageRec`, is used to define how raster data is packaged into a buffer for sending to the output device.

```
struct gxRasterPackageRec {
   Ptr                     bufferSize;
   short                   colorPasses;
   short                    headHeight;
   short                    numberPasses;
   short                    passOffset;
   gxRasterPackageOptions   packageOptions;
};

typedef struct gxRasterPackageRec gxRasterPackageRec,
*gxRasterPackagePtr, **gxRasterPackageHdl;
```

Field descriptions

`bufferSize`	The number of bytes in the raster packaging buffer.
`colorPasses`	The number of print-head passes required to print all colors.
`headHeight`	The height of the print head in pixels.
`numberPasses`	The number of passes required per headheight.
`passOffset`	The offset between passes in pixels.
`packageOptions`	The packaging options, as defined in the next section.

Raster Package Options

The raster package options enumeration defines constants that you can use in the `packageOptions` field of the raster package structure, as described in the previous section.

```
enum {
   gxSendAllColors   = 0x00000001,
   gxInterlaceColor  = 0x00000002,
   gxOverlayColor    = 0x00000004,
   gxUseColor        = (gxInterlaceColor|gxOverlayColor)
};

typedef long gxRasterPackageOptions;
```

Constant descriptions

`gxSendAllColors`	The driver needs to send all bands of data in the raster package, including bands that are all white.
`gxInterlaceColor`	The driver needs to interlace color data to prevent the ribbon from becoming contaminated.
`gxOverlayColor`	The driver can send colors without interlacing because the output device doesn't have a ribbon contamination problem.
`gxUseColor`	The driver needs to send color data. This option implies either `gxInterlaceColor` or `gxOverlayColor`.

Constants and Data Types for the PostScript Imaging System

This section describes the enumerations and structures that are used by the PostScript imaging system messages. You need to know about these data types if you are writing a printing extension or printer driver for a printing device that uses the PostScript imaging system.

PostScript Imaging System Structure

The PostScript imaging system structure, of data type `gxPostScriptImageDataRec`, is allocated and set up by QuickDraw GX during the implementation of the `GXPostScriptGetDocumentProcSetList` and `GXPostScriptDownloadProcSetList` messages, whose descriptions begin on page 4-112. This structure contains information describing the contents of a document that is to be imaged and quantifies the limitations of the device to which the document is to be imaged. QuickDraw GX fills in all the fields in the structure with default values, but your printing extension or printer driver can change any of these fields or use them to derive information about the imaging of the document. For more information about how the fields in this structure are used for color printing, see the section “Color Printing” beginning on page 3-50 in the chapter “Printer Drivers.”

```
struct gxPostScriptImageDataRec {
   short                        languageLevel;
   gxColorSpace                 devCSpace;
   gxColorProfile               devCProfile;
   gxPostScriptRenderOptions    renderOptions;
   long                         pathLimit;
   short                        gsaveLimit;
   short                        opStackLimit;
   scalerStreamTypeFlag         fontType;
   long                         printerVM;
   long                         reserved0;
};

typedef struct gxPostScriptImageDataRec
gxPostScriptImageDataRec, *gxPostScriptImageDataPtr,
**gxPostScriptImageDataHandle;;
```

Field descriptions

`languageLevel`	The language level for the PostScript printer to which your driver is imaging the document.
`devCSpace`	The color space used for this device. The color-space value constants are shown in the chapter “Colors and Color-Related Objects” in *Inside Macintosh: QuickDraw GX Objects.* The values that are valid for this field are `gxGraySpace`, `gxCMYKSpace`, and `gxRGBSpace`.

`devCProfile`	The color profile for this device. Color profiles are explained in the chapter "Colors and Color-Related Objects" in *Inside Macintosh: QuickDraw GX Objects*.
`renderOptions`	Rendering options for the PostScript printer. This value is the combined value of the constants that you include from the PostScript render options enumeration, which is described in the next section.
`pathLimit`	The largest number of points that can be active in the device during the imaging process without generating a PostScript `limitcheck` error, which is described in *PostScript Language Reference Manual*, 2nd Edition.
`gsaveLimit`	The greatest number of gsaves that can be performed on the device without generating a PostScript `limitcheck` error.
`opStackLimit`	The maximum number of objects that can be placed in the stack at any one time without generating a PostScript `limitcheck` error.
`fontType`	The stream type that the imaging system needs to use when downloading a font. The possible values are described in *Inside Macintosh: Typography*.
`printerVM`	The amount of memory available in the printer.

PostScript Render Options

The PostScript render options enumeration defines constants for use in the `renderOptions` field of the PostScript imaging system structure, which is described in the previous section.

```
enum
{
   gxNeedsAsciiOption            = 0x00000001,
   gxNeedsCommentsOption         = 0x00000002,
   gxBoundingBoxesOption         = 0x00000004,
   gxPortablePostScriptOption    = 0x00000008,
   gxUseLevel2ColorOption        = 0x00000080
};

typedef long gxPostScriptRenderOptions;
```

Constant descriptions

`gxNeedsAsciiOption`
: The driver needs to convert all binary data to 7-bit ASCII values.

`gxNeedsCommentsOption`
: The driver needs to issue PostScript comments.

`gxBoundingBoxesOption`
: The driver needs to calculate values for the `%%BoundingBox:` and `%%PageBoundingBox:` variables. This option implies the `gxNeedsCommentsOption` constant.

`gxPortablePostScriptOption`	The driver needs to generate PostScript that is not device specific.
`gxUseLevel2ColorOption`	The driver uses Level 2 device-independent color when printing to a Level 2 output device.

PostScript Glyphs Structure

QuickDraw GX constructs the PostScript glyphs structure, of data type `gxPrinterGlyphsRec`, so a driver can communicate with the imaging system about the fonts and glyphs that are available on a printer. This structure is used by the `GXPostScriptGetPrinterGlyphsInformation` message, which is described on page 4-116.

```
struct gxPrinterGlyphsRec {
   gxFont            theFont;
   long              nGlyphs;
   gxFontPlatform    platform;
   gxFontScript      script;
   gxFontLanguage    language;
   long              vmUsage;
   unsigned long     glyphBits[1];
};

typedef struct gxPrinterGlyphsRec gxPrinterGlyphsRec;
```

Field descriptions

`theFont`	The font that this information is for.
`nGlyphs`	The number of glyphs that the font contains.
`platform`	The platform of the font.
`script`	The script code of the font. This value has no meaning if the value of the `platform` field is -1.
`language`	The language code of the font. This value has no meaning if the value of the `platform` field is -1.
`vmUsage`	The amount of printer memory required for the font.
`glyphBits`	A variable-sized array containing the data for the glyphs. This array must be aligned on a long word boundary. Its size in long words can be computed by the formula: `(nGlyphs + 31) / 32`.

You can read about font glyphs, platforms, scripts, and languages in *Inside Macintosh: QuickDraw GX Typography.*

PostScript Procedure Set List Structure

The PostScript procedure set list structure, of data type `gxProcSetListRec`, contains a list of the procedures needed to image the specified document. This structure is used by the `GXPostScriptGetDocumentProcSetList` and `GXPostScriptDownloadProcSetList` messages, whose descriptions begin on page 4-112.

```
struct gxProcSetListRec {
   Signature   clientId;
   OSType      controlType;
   short       controlid;
   OSType      dataType;
   long        reserved0;
};

typedef struct gxProcSetListRec gxProcSetListRec,
*gxProcSetListPtr, **gxProcSetListHdl;
```

Field descriptions

`clientId`	The unique ID provided by the printing extension or printer driver to avoid resource conflicts.
`controlType`	A resource type that serves as a control resource for the procedure set.
`controlid`	The ID of the resource that controls the downloading of the procedure set.
`dataType`	The resource type that, in combination with the control resource, specifies which resources are to be interpreted as belonging to the procedure set.

PostScript Query Results

The PostScript query results enumeration defines constants that are the possible results of the `GXPostScriptQueryPrinter` message, which is described on page 4-101.

```
enum {
   gxPrinterOK          = 0,
   gxInitializePrinter  = 1,
   gxFilePrinting       = 2,
   gxResetPrinter       = 128
};
```

Constant descriptions

`gxPrinterOK`	The printer responded to the query with enough information to initiate printing of the job.
`gxInitializePrinter`	The printer responded to the query by indicating that it needs to be initialized.
`gxFilePrinting`	The printer responded to the query by indicating that the print job has been redirected to a file.
`gxResetPrinter`	The printer responded to the query by indicating that it needs to be restarted.

Constants and Data Types for the Vector Imaging System

This section describes the enumerations and structures that are used with the vector imaging system messages. You need to know about these data types if you are writing a printing extension or printer driver for a printing device that uses the vector imaging system.

Vector Halftone Structure

The vector halftone structure, of data type `gxVHalftoneRec`, defines the halftone information for a vector device. This structure is used with the vector imaging system structure, which is described on page 4-32.

```
struct gxVHalftoneRec {
   gxColorSpace          halftoneSpace;
   gxHalftoneCompRec     halftoneComps[4];
   long                  penIndexForBW;
};

typedef struct gxVHalftoneRec gxVHalftoneRec;
```

Field descriptions

`halftoneSpace`	The color space for this device. The color-space value constants are shown in the chapter “Colors and Color-Related Objects” in *Inside Macintosh: QuickDraw GX Objects.*
`halftoneComps`	An array of vector halftone component structures. This array has four entries, one for each color component. The vector halftone component structure is described in the next section.
`penIndexForBw`	The pen index for drawing 1-bit deep or black-and-white bitmaps.

Vector Halftone Component Structure

The vector halftone component structure, of data type `gxVHalftoneCompRec`, describes the values to use for creating a halftone for a color component.

```
struct gxVHalftoneCompRec {
   Fixed    angle;
   long     penIndex;
};

typedef struct gxVHalftoneCompRec gxVHalftoneCompRec;
```

Field descriptions

`angle`	The halftone angle for this component.
`penIndex`	The index of the pen to use for drawing this component.

Vector Shape Structure

The vector shape structure, of data type `gxVectorShapeDataRec`, is used with the `GXVectorVectorizeShape` message, which is described on page 4-130.

```
struct gxVectorShapeDataRec {
   gxVectorShapeOptions    shapeOptions;
   long                    maxPolyPoints;
   Fixed                   shapeError;
   Fixed                   textSize;
   Fixed                   frameSize;
};

typedef struct gxVectorShapeDataRec gxVectorShapeDataRec;
```

Field descriptions

`shapeOptions`	Options that you can include for rendering shapes into vectors. This value is the combined value of the constants that you include from the vector shape options enumeration, which is described in the next section.
`maxPolyPoints`	The maximum number of points that a polygon can contain for drawing on this device. If the value of this field is 0, QuickDraw GX converts polygons into vectors rather than sending polygons to the device.
`shapeError`	The maximum number of pixels by which the vectors created to approximate a curve can deviate from the original. The smaller this value, the more vectors are used to approximate each curve, which means more memory, but less deviation in shape. A value of `0x00002000` is typically used, which allows an error of no more than one 1/8 of a pixel at 72 dots per inch.
`textSize`	Any text above this size is filled, while any text below this size is outlined but not filled.
`frameSize`	Any shapes with frames larger than this size are filled, while shapes with any shapes with frames smaller than this size are stroked.

Vector Shape Options

The vector shape options enumeration provides constants that you can use in the `shapeOptions` field of the vector shape structure, which is described in the previous section.

```
enum {
   gxUnidirectionalFill    = 0x00000001,
   gxAlsoOutlineFilledShape= 0x00000002
};

typedef long gxVectorShapeOptions;
```

Constant descriptions

`gxUnidirectionalFill`
: The driver needs to generate scan lines in one direction only. This value is useful for transparencies.

`gxAlsoOutlineFilledShapes`
: The driver also needs to draw the outlines of filled shapes.

Vector Imaging System Structure

The vector imaging system structure, of data type `gxVectorImageDataRec`, is the structure that QuickDraw GX uses to maintain information about the state of the vector imaging sytem. This data type is used with the `GXRenderPage` message, which is described on page 4-130.

```
struct gxVectorImageDataRec {
   gxVectorRenderOptions   renderOptions;
   Fixed                   devRes;
   gxTransform             devTransform;
   gxColorSet              clrSet;
   gxColor                 bgColor;
   gxVHaltoneRec           halftoneInfo;
   gxPenTableHdl           hPenTable;
   gxRectangle             pageRect;
   gxVectorShapeDataRec    shapeData;
};

typedef struct gxVectorImageDataRec gxVectorImageDataRec,
*gxVectorImageDataPtr, **gxVectorImageDataHdl;
```

Field descriptions

`renderOptions`	Options to control the vector rendering. This value is the combined values of the constants that you include from the vector render options enumeration, which is described in the next section.
`devRes`	The resolution of the device.

`devTransform`	The transform for the device.
`clrSet`	The entire set of colors available on the device.
`bgColor`	The background color in the specified color space.
`halftoneInfo`	The halftone information for color bitmaps. This structure is described in the section "Vector Halftone Structure" on page 4-30.
`hPenTable`	The complete list of pens available on the device, along with the thickness and pen position of each. This structure is described in the section "Vector Pen Table Structure" on page 4-34.
`pageRect`	The dimensions of the page. If you set the coordinates of this rectangle to all zeros, the page format values are used.
`shapeData`	The information on how to render a shape on the device.

Vector Render Options

The vector render options enumeration defines constants for use in the `renderOptions` field of the vector imaging system structure.

```
enum {
   gxColorSort       = 0x00000001,
   gxATransferMode   = 0x00000002,
   gxNoOverlap       = 0x00000004,
   gxAColorBitmap    = 0x00000008,
   gxSortbyPenPos    = 0x00000010,
   gxPenLessPlotter  = 0x00000020,
   gxCutterPlotter   = 0x00000040,
   gxNoBackGround    = 0x00000080
};

typedef long gxVectorRenderOptions;
```

Constant descriptions

`gxColorSort`	You need to specify this option for pen plotters.
`gxATransferMode`	The driver needs to resolve transfer modes.
`gxNoOverlap`	The driver needs to produce non-overlapping output.
`gxAColorBitmap`	The driver needs to produce color bitmap output.
`gxSortbyPenPos`	The driver needs to draw shapes in the order of the pens in the pen table (which is not the same as the order of the pens in the pen carousel).
`gxPenLessPlotter`	The output device is a raster printer or plotter.
`gxCutterPlotter`	The output device has a cutter.
`gxNoBackGround`	Shapes that map to the background color are not sent to the driver.

Vector Pen Table Structure

The vector pen table structure, of data type `gxPenTable`, defines the pens that are available for a pen table on a vector device. This structure is used with the vector imaging system structure, which is described on page 4-32, and with the `GXVectorLoadPens` message, which is described on page 4-128.

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

typedef struct gxPenTable gxPenTable, *gxPenTablePtr,
**gxPenTableHdl;
```

Field descriptions

`numPens`	The number of pen table entry structures in the `pens` array.
`pens`	An array of pen table entry structures, with one entry for each pen that is available in this pen table.

Vector Pen Table Entry Structure

The vector pen table entry structure, of data type `gxPenTableEntry`, defines a single pen that is in a pen table used on a vector device.

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

typedef struct gxPenTableEntry gxPenTableEntry;
```

Field descriptions

`penName`	The name of the pen that this structure describes.
`penColor`	The color of this pen. This color matches one of the entries in the color set for this device.
`penThickness`	The size of this pen.
`penUnits`	The units that define the `penThickness` value, as described in the next section.
`penPosition`	The position of this pen in the devices's pen carousel. If the pen is not loaded, it is set to the value `kPenNotLoaded` (-1).

Vector Pen Units

The vector pen units enumeration defines the kind of units used to specify the thickness of a pen table entry, as described in the previous section.

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

Constant descriptions

`gxDeviceUnits`	The pen thickness is specified in device units.
`gxMMUnits`	The pen thickness is specified in millimeters.
`gxInchesUnits`	The pen thickness is specified in inches.

User Interface Constants and Data Types

This section describes the enumerations and structures that QuickDraw GX uses with the dialog box messages. You need to know about these data types when writing the user interface portion of a printing extension or printer driver.

The Panel Information Structure

The panel information structure, of data type `gxPanelInfoRecord`, provides information to the panel about the current dialog box and panel event. This structure is used with the `GXHandlePanelEvent` and `GXFilterPanelEvent` messages, whose descriptions begin on page 4-85.

```
struct gxPanelInfoRecord {
   gxPanelEvent   panelEvt;
   short          panelResId;
   DialogPtr      pDlg;
   EventRecord    *theEvent;
   short          itemHit;
   short          itemCount;
   short          evtAction;
   short          errorStringId;
   gxFormat       theFormat;
   void           *refCon;
};
```

```
typedef struct gxPanelInfoRecord gxPanelInfoRecord;
```

Field descriptions

`panelEvt`	The event to filter or handle.
`panelResId`	The resource ID of the current panel (`'ppnl'`) resource.
`pDlg`	A pointer to the dialog box structure.
`theEvent`	A pointer to the event that occurred.
`itemHit`	The actual item number where the event occurred, using the item-numbering scheme of the Dialog Manager.
`itemCount`	The item count before your panel's items in the dialog box.
`evtAction`	The action that results once this event is processed. This value is one of the constants defined in the panel event actions enumeration, which is described on page 4-38. This field is only meaningful for filtering, and is used for parsing.
`errorStringId`	The ID of the `'STR '` resource to put in the error alert. A value of 0 in this field indicates that there is no error string to display.
`theFormat`	The current format. This is only meaningful in a Custom Page Setup dialog box.
`refcon`	A reference constant for use by the generator of the panel.

Panel Events

The panel event enumeration defines the possible event types that can occur in a panel. This data type is used with the panel information structure, which is described in the previous section.

```
enum {
   gxPanelNoEvt         = (gxPanelEvent) 0,
   gxPanelOpenEvt       = (gxPanelEvent) 1,
   gxPanelCloseEvt      = (gxPanelEvent) 2,
   gxPanelHitEvt        = (gxPanelEvent) 3,
   gxPanelActivateEvt   = (gxPanelEvent) 4,
   gxPanelDeactivateEvt = (gxPanelEvent) 5,
   gxPanelIconFocusEvt  = (gxPanelEvent) 6,
   gxPanelPanelFocusEvt = (gxPanelEvent) 7,
   gxPanelFilterEvt     = (gxPanelEvent) 8,
   gxPanelCancelEvt     = (gxPanelEvent) 9,
   gxPanelConfirmEvt    = (gxPanelEvent) 10,
   gxPanelDialogEvt     = (gxPanelEvent) 11,
   gxPanelOtherEvt      = (gxPanelEvent) 12,
   gxUserWillConfirmEvt = (gxPanelEvent) 13
};

typedef long gxPanelEvent;
```

Constant descriptions

`gxPanelNoEvt`	No event has occurred.
`gxPanelOpenEvt`	The panel is about to open. It needs to be initialized and drawn.
`gxPanelCloseEvt`	The panel is about to close.
`gxPanelHitEvt`	The user has selected an item in the panel.
`gxPanelActivateEvt`	The dialog box in which the panel resides has just been activated.
`gxPanelDeactivateEvt`	The dialog box in which the panel resides is about to be deactivated.
`gxPanelIconFocusEvt`	The focus has changed from the panel to the icon list.
`gxPanelPanelFocusEvt`	The focus has changed from the icon list to the panel.
`gxPanelFilterEvt`	The panel event needs to be filtered.
`gxPanelCancelEvt`	The user has selected the Cancel button in the dialog box.
`gxPanelConfirmEvt`	The user has selected the OK button in the dialog box.
`gxPanelDialogEvt`	An event has occurred in the panel that is going to be handled by a dialog box handler such as the application, a printing extension, a printer driver, or the Macintosh system software.
`gxPanelOtherEvt`	A different kind of event, such as an operating-system event, has occurred in the panel.
`gxPanelUserWillConfirmEvt`	The user has selected the confirm button, which means that it is time to parse panel interdependencies.

Panel Responses

A handler of a panel in a dialog box (including applications, printing extensions, printer drivers, and Macintosh system software) can return any value of type `OSErr` as the result of handling the panel. In addition, a panel handler can return an event of type `gxPanelResult`, as shown here. This data type is used with the `GXHandlePanelEvent` message, which is described on page 4-85.

```
enum {
   gxPanelNoResult            = 0,
   gxPanelCancelConfirmation  = 1
};

typedef long gxPanelResult;
```

Constant descriptions

`gxPanelNoResult`
: The result field does not currently have any meaning.

`gxPanelCancelConfirmation`
: This result is only valid if the panel event (as described in the previous section) was of type `gxPanelUserWillConfirmEvt`. After the user confirms the panel, if the panel handler discovers that the user entered an inappropriate value, the panel handler alerts the user to the problem and generates this response, which tells QuickDraw GX to not confirm the dialog box. This allows the user the opportunity to fix the problem.

Panel Event Actions

The panel event actions enumeration defines the constants used in the `evtAction` field of the panel information structure, which is described on page 4-35. Each value defines what action takes place after an event is processed.

```
enum {
   gxOtherAction      = 0,
   gxClosePanelAction = 1,
   gxCancelDialogAction = 2,
   gxConfirmDialogAction= 3
};
```

Constant descriptions

`gxOtherAction` The current item does not change after processing this event.

`gxClosePanelAction`
: The panel is closed after this event is processed.

`gxCancelDialogAction`
: The dialog box is canceled after this event is processed.

`gxConfirmDialogAction`
: The dialog box is confirmed after this event is processed.

Parse Range Results

The parse range results enumeration provides the constants that are used as the return result of the `GXParsePageRange` message, which is described on page 4-60.

```
enum {
   gxRangeNotParsed     = (gxParsePageRangeResult) 0,
   gxRangeParsed        = (gxParsePageRangeResult) 1,
   gxRangeBadFromValue  = (gxParsePageRangeResult) 2,
   gxRangeBadToValue    = (gxParsePageRangeResult) 3
};
typedef long gxParsePageRangeResult;
```

Constant descriptions

`gxRangeNotParsed`	QuickDraw GX has not yet parsed a page range in the string.
`gxRangeParsed`	QuickDraw GX has successfully parsed a page range in the string.
`gxRangeBadFromValue`	QuickDraw GX has encountered an invalid value in the "from page" string during the parse.
`gxRangeBadToValue`	QuickDraw GX has encountered an invalid value in the "to page" string during the parse.

The Status Structure

The status structure, of data type `gxStatusRecord`, contains status information used with the status messages `GXJobStatus`, `GXWriteStatusToDTPWindow`, `GXInitializeStatusAlert`, `GXHandleAlertEvent`, and `GXHandleAlertStatus`. These messages are described in the section "Printing Messages Reference" beginning on page 4-9.

```
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

`statusType`	The type of status that this structure represents. This is one of the values shown in Table 4-1.
`statusId`	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.
`statusAlertId`	The ID of the printing alert for this status.
`statusOwner`	The creator type of the owner of this status structure.
`statResId`	The resource ID for the status (`'stat'`) resource used to process this status.
`statResIndex`	The index value for indexing into the status resource for this status.
`dialogResult`	The ID of the button string that was selected to dismiss the printing alert box associated with this status.

`bufferLen`	The number of bytes in the status buffer.
`statusBuffer`	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 record. u

Table 4-1 shows the status types that you can specify in a status record.

Table 4-1 Status type IDs

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

The Manual Feed Structure

The manual feed structure, of data type `gxManualFeedRecord`, is passed in the `statusBuffer` field of the status structure for a manual feed alert.

```
struct gxManualFeedRecord {
   Boolean  canAutoFeed;
   Str31    paperTypeName;
};

typedef struct gxManualFeedRecord gxManualFeedRecord;
```

Field descriptions

`canAutoFeed`	A Boolean value that is `true` when the driver can handle swtiching into automatic paper-feeding mode from manual-feed mode in the middle of a print job. This field is otherwise `false`.
`paperTypeName`	The string name of the paper type that is to be fed manually. This field can also be an array of paper-type names.

The Display Structure

The display structure, of data type `gxDisplayRecord`, is used with the `GXWriteStatusToDTPWindow` message to send status information for display in the desktop printer window. The `GXWriteStatusToDTPWindow` message is described on page 4-163.

```
struct gxDisplayRecord {
   Boolean  useText;
   Handle   hPicture;
   Str255   theText;
};

typedef struct gxDisplayRecord gxDisplayRecord;
```

Field descriptions

`useText`	A Boolean value that specifies whether to use the text supplied in the `theText` field. If `true`, the text in the `theText` field is displayed; if `false`, the picture referred to by the `hPicture` field is displayed.
`hPicture`	A handle to the picture data to display. QuickDraw GX does not yet support this feature.
`theText`	The text string to display.

Note

The `hPicture` field is not yet used. At this time, you must set the value of `useText` to `true`. u

Printing Messages

This section describes the messages that you can override to develop a printing extension or printer driver. This section presents these messages in categories that relate to the tasks for which you use each message.

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

A number of the printing messages can return communications result codes, which are listed in Table 4-2. The result codes section for each of these messages refers to this table.

Table 4-2 Communications errors returned by many of the printing messages

Result code	Explanation
`gxAioTimeout`	The operation timed out
`gxAioBdRqstState`	Asynchronous communications are in an unexpected state for the operation
`gxAioBadConn`	The I/O connection reference number is not valid
`gxAIOInvalidXfer`	Bad values were discovered in the data transfer structure
`gxAioNoRqstBlks`	No request blocks are available to process the request
`gxAioNoDataXfer`	There is no pointer to a data transfer structure specified
`gxAioTooManyAutos`	Automatic status request is already active
`gxAioNoAutoStat`	The connection is not configured for automatic status request
`gxAioBadRqstID`	The I/O request identifier is not valid
`gxAioCantKill`	The communications protocol does not support I/O termination
`gxAioAlreadyExists`	The protocol-specific data was already specified
`gxAioCantFind`	The protocol-specific data was not found
`gxAioDeviceDisconn`	The printer was disconnected from the computer
`gxAioNotImplemented`	The specified function is not implemented
`gxAioOpenPending`	There is already a connection opened
`gxAioNoProcotolData`	There was not any protocol-specific data specified in the request
`gxAioRqstKilled`	The I/O request was terminated
`gxBadBaudRate`	The specified baud rate is not valid
`gxBadParity`	The specified parity value is not valid
`gxBadStopBits`	The specified stop-bits value is not valid
`gxBadDataBits`	The specified data-bits value is not valid
`gxBadPrinterName`	The specified printer name is not valid
`gxAioBadMsgType`	The message type specified in the data transfer structure is not valid
`gxAioCantFindDevice`	The target device could not be located
`gxAioOutOfSeq`	A non-atomic SCSI request was submitted in the wrong order

You must define each of your message overrides with the interface that is shown in each message description. Each override must match the formal declaration of the message that it is overriding: it must take the same parameters, in the same order, as shown in the example. You can provide your own name (whatever name you like) for each override that you define.

For clarity, Apple printing extensions and printer drivers apply the same prefix to each override they include. For example, the Apple Personal LaserWriter SC printer driver provides overrides named `SD_ShutDown`, `SD_DefaultPrinter`, `SD_DespoolPage`, and so on. The background picture printing extension includes overrides named `BWInitialize`, `BWShutdown`, and `BWDespoolPage`. You can choose to follow a similar convention for naming your message override functions. For example, you could name your override of the `GXCountPages` messages `MyCountPages`.

Most of the message descriptions also include a “Special Considerations” section, which describes whether you can send the message yourself and whether you can totally override the message. For messages that you can partially override, this section includes information about whether to forward the message prior to or after performing your own tasks.

Storage Messages

Some messages are useful in all phases of printing and are needed by all message handlers. You can use these common messages to set up a “global” context for your handler. They allow you to save data between messages when needed.

GXInitialize

QuickDraw GX sends the `GXInitialize` message at the start of a new printing job. You can override the `GXInitialize` message to perform any initialization your message handler needs to carry out its tasks. Your override of the `GXInitialize` message must match the following formal declaration:

```
OSErr MyInitialize (void);
```

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

DESCRIPTION

QuickDraw GX sends the `GXInitialize` message at the start of a new printing job, just after the application has called the `GXNewJob` function.

You can override the `GXInitialize` message to perform any necessary initialization and to allocate storage for your extension or driver. `GXInitialize` is the first message a message handler receives after it is loaded into the message chain. If you need to allocate storage for global data in your override of `GXInitialize`, you need to call the `NewHandle` function and store the handle by calling the `SetMessageHandlerInstanceContext` function. Or you can call the `NewMessageGlobals` function to set up a globals environment to access so that you can access the global data. This stores the handle into the context for any other messages that you override. .

The default implementation of the `GXInitialize` message allocates memory needed by QuickDraw GX in its default implementation of the printing messages.

SPECIAL CONSIDERATIONS

You never send the `GXInitialize` message yourself.

You never forward the `GXInitialize` message.

RESULT CODES

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

SEE ALSO

You can find examples of overrides of the `GXInitialize` message in Listing 2-6 on page 2-16 in the chapter “Printing Extensions” and in Listing 3-4 on page 3-24 in the chapter “Printer Drivers.”

The `GXNewJob` function is described in *Inside Macintosh: QuickDraw GX Printing.*

The `NewHandle` function is described in *Inside Macintosh: Memory.* The `NewMessageGlobals` function is described in the chapter “Message Manager” in *Inside Macintosh: QuickDraw GX Environment and Utilities.*

The `SetMessageHandlerInstanceContext` function is described in *Inside Macintosh: Processes.*

GXShutDown

QuickDraw GX sends the `GXShutDown` message at the end of a print job. You can override the `GXShutDown` message to clean up or dispose of any private storage you allocated in your override of the `GXInitialize` message. Your override of the `Shutdown` message must match the following formal declaration:

```
OSErr MyShutDown (void);
```

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

DESCRIPTION

QuickDraw GX sends the `GXShutDown` message after a print job is complete. The printing context still exists when this message is sent.

You need to override this message to deallocate any storage you were using within the context of the print job and deallocate the handle you placed into the context as a result of the `GXInitialize` message. The `GXShutDown` message is always the last message a message handler receives before it is removed from the message chain.

The default implementation of the `GXShutDown` message deallocates memory that was allocated by the printing system.

SPECIAL CONSIDERATIONS

You need to override the `GXShutDown` message and deallocate storage if you have also overridden the `GXInitialize` message.

You never send the `GXShutDown` message yourself.

You never forward the `GXShutDown` message to other message handlers.

RESULT CODES

`gxSegmentLoadFailedErr`	A required code segment could not be found, or there was not enough memory to load it.
`gxPrUserAbortErr`	The user has canceled printing.
`memFullErr`	There is not enough memory to retrieve the tagged data.
`resNotFound`	The specified data could not be found.

SEE ALSO

You can find examples of overrides of the `GXShutDown` message in Listing 2-18 on page 2-31 in the chapter "Printing Extensions" and in Listing 3-18 on page 3-48 in the chapter "Printer Drivers."

The `GXInitialize` message is described in the previous section.

GXFetchTaggedData

QuickDraw GX, an application, a printing extension, or a printer driver can send the `GXFetchTaggedData` message to retrieve or modify data from a resource. You can override the `GXFetchTaggedData` message to modify data from your resources at run time. Use `GXFetchTaggedData` to retrieve and modify resource data that you access by resource type and ID. Your override of the `GXFetchTaggedData` message must match the following formal declaration:

```
OSErr MyFetchTaggedData (ResType aResType, short id,
                         Handle *aHandle, Signature owner);
```

`aResType`	On input, the resource type in which you are interested.
`id`	The resource ID.
`aHandle`	On return, a pointer to a handle that contains the resource data.
`owner`	The owner of the requested data.

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

DESCRIPTION

QuickDraw GX or any message handler can send this message to obtain resource information from a printer driver. You can send this message to yourself to obtain data from a particular resource.

You specify the resource type and ID of the resource to retrieve in the `aResType` and `id` parameters, respectively. The resource handle is returned with a handle value in the `aHandle` parameter. The `owner` parameter defines the owner type of the resource.

The default implementation of this message accesses the resource that belongs to the driver by calling the `GetResource` function and then calling the `DetachResource` function. Override this message if you wish to change resource data that is being read from a printing extension, a printer driver, or the desktop printer.

If you are not the owner of the data (that is, if the owner ID is not your creator type), you need to forward this message. If the owner ID is `'drvr'`, the message is intended for the currently active driver. If you are the owner, you need to create a handle for the tagged data and fill it out. If you use the `GetResource` function to get the handle, be sure to call the `DetachResource` function to change the handle into a non-Resource Manager handle.

RESULT CODES

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

SEE ALSO

The `GetResource` and `DetachResource` functions are described in the chapter “Resource Manager” in *Inside Macintosh: More Macintosh Toolbox*.

Print Object Messages

Print object messages allow you to manipulate the various print objects that are created and referenced by an application during all phases of the printing process. These objects are the job object, the format object, and the paper-type object. You can choose to override print object messages to add additional data to these objects. You can also change the default information to something that has more meaning for you.

GXDefaultJob

QuickDraw GX sends the `GXDefaultJob` message when an application creates a new job. You can override the `GXDefaultJob` message to add to or modify the contents of the job object. Your override of the `GXDefaultJob` message must match the following formal declaration:

```
OSErr MyDefaultJob (void);
```

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

DESCRIPTION

QuickDraw GX sends the `GXDefaultJob` message when an application calls the `GXNewJob` function to create a new job object.

You override this message if you need to add additional data to the job object when it is created. After you forward this message down the message chain, you can add your own information to the job object or change information that was placed there by the default implementation.

The default implementation of this message provides the default information for this job object.

SPECIAL CONSIDERATIONS

You can find an example of an override of the `GXDefaultJob` message in Listing 3-6 on page 3-28 in the chapter "Printer Drivers."

You never send the `GXDefaultJob` message yourself.

You must forward the `GXDefaultJob` message to other message handlers. Always forward it before you add or change job information.

RESULT CODES

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

SEE ALSO

The `GXNewJob` function is described in *Inside Macintosh: QuickDraw GX Printing*.

GXDefaultFormat

QuickDraw GX sends the `GXDefaultFormat` message when an application creates a new format object. You can override the `GXDefaultFormat` message to modify or add to the contents of a newly created format object. Your override of the `GXDefaultFormat` message must match the following formal declaration:

```
OSErr MyDefaultFormat (gxFormat aFormat);
```

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

DESCRIPTION

QuickDraw GX sends the `GXDefaultFormat` message when an application calls the `GXNewFormat` function to create a new format.

You override this message if you need to add additional data to the format object when it is created. After you forward this message down the message chain, you can add your own information to the format object or change information that was placed there by the default implementation. The default implementation of this message provides the default information for this format object.

SPECIAL CONSIDERATIONS

You never send the `GXDefaultFormat` message yourself.

You must forward the `GXDefaultFormat` message to other message handlers. Always forward it before you add or change format information.

RESULT CODES

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

SEE ALSO

The `GXNewFormat` function is described in *Inside Macintosh: QuickDraw GX Printing.*

GXDefaultPaperType

QuickDraw GX sends the `GXDefaultPaperType` message when an application creates a new paper-type object. You can override the `GXDefaultPaperType` message to make changes in the paper type used by a printing job. Your override of the `GXDefaultPaperType` message must match the following formal declaration:

```
OSErr MyDefaultPaperType (gxPaperType aPaperType);
```

`aPaperType`	A paper-type object to fill in with the default value.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXDefaultPaperType` message when an application calls the `GXNewPaperType` function to create a new paper type.

You override this message if you need to add additional data to the paper-type object at the time that it is created or if you wish to change the default paper type to another type. After you forward this message down the message chain, you can add your own information to the paper-type object or change information that was placed there by the default implementation. The default implementation of this message provides the default information for this paper type, including size, margins, and PostScript data.

SPECIAL CONSIDERATIONS

You never send the `GXDefaultPaperType` message yourself.

You must forward the `GXDefaultPaperType` message to other message handlers. Always forward it before you add or change paper-type information.

RESULT CODES

`gxSegmentLoadFailedErr`	A required code segment could not be found, or there was not enough memory to load it.
`gxPrUserAbortErr`	The user has canceled printing.
`gxPaperTypeNotFound`	The specified paper type could not be found.
`gxNoSuchPTGroup`	The specified paper type could not be found.

SEE ALSO

The `GXNewPaperType` function is described in *Inside Macintosh: QuickDraw GX Printing*.

GXDefaultPrinter

QuickDraw GX sends the `GXDefaultPrinter` message when an application creates a new printer object. You can override the `GXDefaultPrinter` message to modify the default printer object as it is being created. Your override of the `GXDefaultPrinter` message must match the following formal declaration:

```
OSErr MyDefaultPrinter (gxPrinter aPrinter);
```

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

DESCRIPTION

QuickDraw GX sends the `GXDefaultPrinter` message when an application calls the `GXNewJob` function to create a new job object.

You override this message if you need to modify the default printer object at the time that it is created. For example, you may want your printer driver to supply an application with the list of the valid color spaces and printing devices that it supports. After you forward this message down the message chain, you can add your own information to the printer object or change information that was placed there by the default implementation. The default implementation of this message provides the default information for the printer object.

SPECIAL CONSIDERATIONS

You never send the `GXDefaultPrinter` message yourself.

You must forward the `GXDefaultPrinter` message to other message handlers. Always forward it before you add or change printer information.

RESULT CODES

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

SEE ALSO

You can find an example of an override of the `GXDefaultPrinter` message in Listing 3-5 on page 3-25 in the chapter "Printer Drivers."

The `GXNewJob` function is described in *Inside Macintosh: QuickDraw GX Printing*.

GXDefaultDesktopPrinter

QuickDraw GX sends the `GXDefaultDesktopPrinter` message when the user creates a new desktop printer with the Chooser. You can override the `GXDefaultDesktopPrinter` message to modify the configuration file of the desktop printer. Your override of the `GXDefaultDesktopPrinter` message must match the following formal declaration:

```
OSErr MyDefaultDesktopPrinter (Str31 dtpName);
```

`dtpName`	The name of the desktop printer.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXDefaultDesktopPrinter` message when a user creates a new desktop printer.

You override this message if you need to add additional data to the printer's configuration file. For example, you may want your driver to supply an application with the list of the valid color spaces and printing devices that it supports or you might want to fill in the default configuration for the printer. After you forward this message down the message chain, you can add your own information to the printer object or change information that was placed there by the default implementation. The default implementation of this message fills in the default information for the printer object.

SPECIAL CONSIDERATIONS

You never send the `GXDefaultDesktopPrinter` message yourself.

You must forward the `GXDefaultDesktopPrinter` message to other message handlers. Always forward it before you add or change configuration information.

RESULT CODES

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

SEE ALSO

Creating desktop printers is described in *Inside Macintosh: QuickDraw GX Printing*.

Application Messages

QuickDraw GX sends application messages in response to an application calling QuickDraw GX functions to accomplish printing-related tasks. You rarely need to override these messages because modifying the data prior to spooling is more easily done by overriding spooling messages, which are described in the section "Spooling Messages" beginning on page 4-67.

GXStartJob

QuickDraw GX sends the `GXStartJob` message when the spooling of a document is initiated. You can override the `GXStartJob` message to set initial values at the start of printing a document. Your override of the `GXStartJob` message must match the following formal declaration:

```
OSErr MyStartJob (StringPtr docName, long pageCount);
```

`docName`	The document name.
`pageCount`	The number of pages in the document.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXStartJob` message to initiate spooling when an application calls the `GXStartJob` function to start printing.

You need to override this message if you want to initialize information when an application begins printing a document. You can also override this message to determine when a document is being spooled.

The default implementation of `GXStartJob` begins the process of saving the document into a spool file. It sends spooling messages to accomplish this.

SPECIAL CONSIDERATIONS

You never send this message yourself.

You must forward the `GXStartJob` message to other message handlers so that they can override it. If your override fails, you need to call the `GXCleanupStartJob` function to notify other handlers of the failure. If another handler returns an error, you must undo anything that you've done and return the same error.

RESULT CODES

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

SEE ALSO

The `GXCleanupStartJob` function is described on page 5-36 in the chapter "Printing Functions for Message Overrides."

GXCleanupStartJob

QuickDraw GX sends the `GXCleanupStartJob` message if a message handler fails during the processing of a `GXStartJob` message. You can override the `GXCleanupStartJob` message to deallocate any storage you allocated in your override of the `GXStartJob` message. Your override code for `GXCleanupStartJob` has no parameters and returns no value. For example, you could declare your override for the `GXCleanupStartJob` message as follows:

```
void MyCleanupStartJob (void);
```

DESCRIPTION

QuickDraw GX sends the `GXCleanupStartJob` message after a message handler calls the `GXCleanupStartJob` function.

You need to override this message to deallocate any storage that you allocated in your override of the `GXStartJob` message.

The default implementation of this message disposes of memory that was allocated for the printing job.

SPECIAL CONSIDERATIONS

You never send the `GXCleanupStartJob` message yourself; however, you can call the `GXCleanupStartJob` function, which then sends this message.

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

SEE ALSO

The `GXStartJob` message is described in the previous section.

The `GXCleanupStartJob` function is described on page 5-36 in the chapter "Printing Functions for Message Overrides."

GXFinishJob

QuickDraw GX sends the `GXFinishJob` message when the spooling of a document is finished. You can override the `GXFinishJob` message to perform operations required by your printing extension or printer driver at the completion of spooling a document. Your override of the `GXFinishJob` message must match the following formal declaration:

```
OSErr MyFinishJob (void);
```

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

DESCRIPTION

QuickDraw GX sends the `GXFinishJob` message when an application calls the `GXFinishJob` function to indicate that spooling of a document is complete.

You need to override this message if you want to know when a document has finished spooling or to finalize information when spooling of a document is done.

The default implementation of `GXFinishJob` completes the spooling process by sending spooling messages. It also updates the job object data to contain the correct number of pages that are to be printed. It sends spooling messages to accomplish this.

SPECIAL CONSIDERATIONS

You never send the `GXFinishJob` message yourself.

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

RESULT CODES

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

SEE ALSO

The `GXFinishJob` function is described in *Inside Macintosh: QuickDraw GX Printing.*

GXJobIdle

QuickDraw GX sends the `GXJobIdle` message when a printer driver, a printing extension, or an application calls the `GXJobIdle` function to release idle time. You can override the `GXJobIdle` message to obtain time from the idle procedure. Your override of the `GXJobIdle` message must match the following formal declaration:

```
OSErr MyJobIdle (void)
```

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

DESCRIPTION

You can override this message to perform tasks during idle time. The default implementation of this message gives up time to other processes.

SPECIAL CONSIDERATIONS

You never send the `GXJobIdle` message yourself; however, you can call the `GXJobIdle` function, which then sends this message.

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

RESULT CODES

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

SEE ALSO

The `GXJobIdle` function is described on page 5-33 in the chapter "Printer Functions for Message Overrides."

GXStartPage

QuickDraw GX sends the `GXStartPage` message to start a new page in the spool file. You can override the `GXStartPage` message to perform any initialization your printing extension or printer driver requires before printing each page. Your override of the `GXStartPage` message must match the following formal declaration:

```
OSErr MyStartPage (gxFormat aFormat, long numViewPorts,
                   gxViewPort *viewPortList);
```

`aFormat`	The format object for the page.

`numViewPorts`
The number of view ports pointed to by the `viewPortList` parameter.

`viewPortList`
A pointer to a list of view ports to use to capture shapes.

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

DESCRIPTION

QuickDraw GX sends this message when an application calls the `GXStartPage` function to start a new page. The application calls the `GXStartPage` and `GXFinishPage` functions once for each page, which causes QuickDraw GX to send the `GXStartPage` and `GXFinishPage` messages. The `GXStartPage` message begins a new page in the spool file and prepares to capture all data drawn to the view ports in the `viewPortList` parameter so that this data can be redirected to the new page.

You need to override this message if you want to initialize information when an application begins printing each page. You can also override this message to determine when a new page is being spooled.

The default implementation of `GXStartPage` installs view port filters on the requested view ports to begin capturing graphics objects.

SPECIAL CONSIDERATIONS

You never send the `GXStartPage` message yourself.

You must forward the `GXStartPage` message to other message handlers so that they can override it. If your override fails, you need to call the `GXCleanupStartPage` function to notify other handlers of the failure. If another handler returns an error, you must undo anything that you've done and return the same error.

RESULT CODES

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

SEE ALSO

The `GXStartPage` and `GxFinishPage` functions are described in *Inside Macintosh: QuickDraw GX Printing.*

The `GXCleanupStartPage` function is described on page 5-37 in the chapter "Printing Functions for Message Overrides."

GXCleanupStartPage

QuickDraw GX sends the `GXCleanupStartPage` message if a message handler fails during the processing of a `GXStartPage` message. You can override the `GXCleanupStartPage` message to deallocate any storage allocated during processing of the `GXStartPage` message. Your override of the `GXCleanupStartPage` message must match the following formal declaration:

```
void MyCleanupStartPage (void);
```

DESCRIPTION

QuickDraw GX sends the `GXCleanupStartPage` message after it receives a `GXCleanupStartPage` function call from a message handler that failed during a `GXStartPage` override and after forwarding the `GXStartPage` message.

You need to override the `GXCleanupStartPage` message if you perform an operation that must be undone within the `GXStartPage` message override. For example, your attempt to allocate memory or initialize a device after forwarding the `GXStartPage` message may fail. When you receive this message, you typically deallocate any storage you were using because of `GXStartPage`.

The default implementation of `GXCleanupStartPage` disposes of the memory that was allocated during the default implementation of the `GXStartPage` message.

SPECIAL CONSIDERATIONS

You never send the `GXCleanupStartPage` message yourself; however, you can call the `GXCleanupStartPage` function, which then sends this message.

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

SEE ALSO

The `GXStartPage` message is described in the previous section.

The `GXCleanupStartPage` function is described on page 5-37 in the chapter "Printing Functions for Message Overrides."

GXFinishPage

QuickDraw GX sends the `GXFinishPage` message when spooling of a page is finished. You can override the `GXFinishPage` message to perform any action required at the end of each page. Your override of the `GXFinishPage` message must match the following formal declaration:

```
OSErr MyFinishPage (void);
```

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

DESCRIPTION

QuickDraw GX sends this message when an application calls the `GXFinishPage` function to complete a page.

You need to override this message if you want to perform an action at the end of each page. QuickDraw GX sends the `GXFinishPage` message once for each `GXStartPage` message that is sent. This message indicates that the data has been drawn and that it can be written to the spool file.

The default implementation of `GXFinishPage` completes the associated page and spools the data for the page by sending the `GXSpoolPage` message. It also releases the view ports that were captured when the `GXStartPage` message was sent.

SPECIAL CONSIDERATIONS

You never send the `GXFinishPage` message yourself.

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

RESULT CODES

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

SEE ALSO

The `GXFinishPage` function is described in *Inside Macintosh: QuickDraw GX Printing*.

GXPrintPage

QuickDraw GX sends the `GXPrintPage` message when the data for a page is ready to be spooled. You can override the `GXPrintPage` message to modify data sent by an application for printing. Your override of the `GXPrintPage` message must match the following formal declaration:

```
void MyPrintPage (gxFormat aFormat, gxShape aPage);
```

`aFormat`	The format object for the page.
`aPage`	The data that belongs on the page in the form of a picture shape.

DESCRIPTION

QuickDraw GX sends this message when an application call the `GXPrintPage` function to print a page.

You rarely need to override this message. If you wish to modify the data that the application sends, it is preferable to override the `GXSpoolPage` message. This works for applications that use `GXPrintPage` for printing and also for those that use the `GXStartPage`/`GXDrawShape`/`GXFinishPage` sequence for printing.

SPECIAL CONSIDERATIONS

You never send the `GXPrintPage` message yourself.

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

SEE ALSO

The `GXPrintPage` function is described in *Inside Macintosh: QuickDraw GX Printing.*

GXJobFormatModeQuery

QuickDraw GX sends the `GXJobFormatModeQuery` message when the application requests or sets format mode information. You can override the `GXJobFormatModeQuery` message to provide support for direct modes. Your override of the `GXJobFormatModeQuery` message must match the following formal declaration:

```
void MyJobFormatModeQuery (gxQueryType aQueryType,
                           void *srcData, void *dstData);
```

`aQueryType`	The type of query being performed.
`srcData`	A pointer to the input data for the query.
`dstData`	A pointer to the output (result) data for the query.

DESCRIPTION

QuickDraw GX sends the `GXJobFormatModeQuery` message when an application calls the `GXJobFormatModeQuery` function to get or set additional format-mode information.

The type of query made by the application is specified in the `aQueryType` parameter, the possible values of which are listed and described in the “Advanced Printing Features” chapter in *Inside Macintosh: QuickDraw GX Printing.* Use of the `srcData` and `dstData` parameters is described in the same chapter.

You only need to override this message if you support job format modes. Based on the query from the application, you provide the requested information.

The default implementation of the `GXJobFormatModeQuery` message does nothing.

SPECIAL CONSIDERATIONS

You never send the `GXJobFormatModeQuery` message yourself.

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

SEE ALSO

You can find an example of an overide of the `GXJobFormatModeQuery` message in Listing 3-10 on page 3-34 in the chapter "Printer Drivers."

The `GXJobFormatModeQuery` function, including the types of queries that you can make and the use of the `srcData` and `dstData` parameters, is described in *Inside Macintosh: QuickDraw GX Printing*.

GXParsePageRange

QuickDraw GX sends the `GXParsePageRange` message when a user selects a range of pages for printing. You can override the `GXParsePageRange` message to validate a page range. Your override of the `GXParsePageRange` message must match the following formal declaration:

```
OSErr MyParsePageRange (StringPtr fromString, StringPtr toString,
                        gxParsePageRangeResult *result);
```

`fromString`	A pointer to a string representation of the from-page.
`toString`	A pointer to a string representation of the to-page.
`result`	On return, a value that specifies the result code for the range parsing. The constants for this value are given in the section "Parse Range Results" on page 4-38.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXParsePageRange` message to validate that a page range entered by the user is appropriate for the print job.

SPECIAL CONSIDERATIONS

You rarely send the `GXParsePageRange` message yourself.

You must always forward the `GXParsePageRange` message to other message handlers.

RESULT CODES

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

Paper-Handling Messages

QuickDraw GX provides the `GXDoesPaperFit` message to determine if a specific paper type can be used on a printer. QuickDraw GX also provides several paper-handling functions, each of which is described in the chapter "Printing Functions for Message Overrides" in this book.

GXDoesPaperFit

QuickDraw GX sends the `GXDoesPaperFit` message to determine if a specific paper type is acceptable on the printing device that your driver supports. You can override the `GXDoesPaperFit` message to set or modify the returned value. Your override of the `GXDoesPaperFit` message must match the following formal declaration:

```
OSErr MyDoesPaperFit (gxTrayIndex whichTray, gxPaperType paper,
                      Boolean *fits);
```

`whichTray`	The tray in which the paper is to be loaded.
`paper`	The paper type.
`fits`	On return, a Boolean value that is `true` if the paper can be used on the printer and `false` if not.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXDoesPaperFit` message to limit the number of paper-type names that appear in the Paper pop-up menu in the Page Setup dialog box. You can override the `GXDoesPaperFit` message to notify QuickDraw GX that a certain paper type does not work in the specified paper tray on your device.

RESULT CODES

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

Color Profile Messages

QuickDraw GX sends color profile messages to allow printing extensions and printer drivers to work with the color profiles that are used for color matching. Color matching and the ColorSync Manager are described in *Inside Macintosh: Advanced Color Imaging*.

GXFindPrinterProfile

QuickDraw GX sends the `GXFindPrinterProfile` message to allow you to modify color-matching information for your printer driver. Your override of the `GXFindPrinterProfile` message must match the following formal declaration:

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

`thePrinter`
: 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.

`returnedProfile`
: On return, a list of color profiles matching the criteria specified by the `searchData` and `index` parameters. If no color profiles are found, this value 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

When an application calls the `GXFindPrinterProfile` function to search for a color profile that matches certain specifications, QuickDraw GX sends the `GXFindPrinterProfile` message to allow your printing extension or printer driver to modify the search criteria or results.

The default implementation of the `GXFindPrinterProfile` message checks the profile that is associated with the view device of the current printer. If no profile match is made, it then checks the profile that is associated with the color profile (`'prof'`) resource that is stored with the desktop printer. If no profile is matched, it then checks the profile that is associated with the color profile resource that is stored with the printer driver. If no match is made, then `nil` is returned in the `returnedProfile` parameter. You can override this message to change the profile matching data prior to forwarding the message or to modify the match results after forwarding the message.

RESULT CODES

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

SEE ALSO

The `GXFindPrinterProfile` function is described in *Inside Macintosh: QuickDraw GX Printing*.

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

GXFindFormatProfile

QuickDraw GX sends the `GXFindFormatProfile` message to allow you to modify color matching information for a specific format object. This message is similar to the `GXFindPrinterProfile` message (described above), except that it finds a color profile that is associated with a format object rather than a printer object. Your override of the `GXFindFormatProfile` message must match the following formal declaration:

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

`theFormat`	The format 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.
`returnedProfile`	On return, a color profile matching the criteria specified by the `searchData` and `index` parameters. If no color profiles are found, this value 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

When an application calls the `GXFindFormatProfile` function to search for a color profile that matches certain specficiations, QuickDraw GX sends the `GXFindFormatProfile` message to allow your printing extension or printer driver to modify the search criteria or results.

The default implementation of the `GXFindFormatProfile` message checks the profile that is associated with the specified format object. If no profile match is made, it then checks the profile that is associated with the color profile (`'prof'`) resource that is stored with the desktop printer. If no profile is match, it then checks the profile that is associated with the color profile resource that is stored with the printer driver. If no match is made, then `nil` is returned in the `returnedProfile` parameter. You can override this message to change the profile matching data prior to forwarding the message or to modify the match results after forwarding the message.

RESULT CODES

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

SEE ALSO

The `GXFindFormatProfile` function is described in *Inside Macintosh: QuickDraw GX Printing.*

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

GXSetPrinterProfile

QuickDraw GX sends the `GXSetPrinterProfile` message to change the current color profile for a printer. Your override of the `GXSetPrinterProfile` message must match the following formal declaration:

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

`thePrinter`	The printer object.
`oldProfile`	The profile that has been associated with the printer object.
`newProfile`	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 override the `GXSetPrinterProfile` message 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.

The values of the `oldProfile` and `newProfile` parameters define what happens in response to this message, as shown in Table 4-3.

Table 4-3 The actions of the `GXSetPrinterProfile` message

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

The default implementation of this message modifies the list of profiles that are associated with the printer object, as shown in Table 4-3.

RESULT CODES

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

SEE ALSO

The `GXSetPrinterProfile` function is described in *Inside Macintosh: QuickDraw GX Printing*.

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

GXSetFormatProfile

QuickDraw GX sends the `GXSetFormatProfile` message to change the current color profile for a format object. Your override of the `GXSetFormatProfile` message must match the following formal declaration:

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

`theFormat`	The format object.
`oldProfile`	The profile that has been associated with the format object.
`newProfile`	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 override the `GXSetFormatProfile` message 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.

The values of the `oldProfile` and `newProfile` parameters define what happens in response to this message, as shown in Table 4-3.

Table 4-4 The actions of the `GXSetFormatProfile` message

Value of `oldProfile`	Value of `newProfile`	Action taken
`nil`	`nil`	None
Valid	`nil`	`oldProfile` is deleted from the list of profiles associated with the format object
`nil`	Valid	`newProfile` is added to the list of profiles for the format 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 format object

The default implementation of this message modifies the list of profiles that are associated with the format object, as shown in Table 4-3.

RESULT CODES

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

SEE ALSO

The `GXSetFormatProfile` function is described in *Inside Macintosh: QuickDraw GX Printing*.

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

Spooling Messages

When an application sends shapes to be printed, QuickDraw GX uses spooling messages to send these shapes to spool files. It sends these messages when the application is calling QuickDraw GX functions to accomplish printing-related tasks. You can override spooling messages to change the data as it is being written to disk.

Some printing extensions and printer drivers need to perform their own spooling or create a custom format for the spool file. If this is the case for your extension or driver, you need to override all of the spooling and despooling messages.

When you override a spooling message, you need to choose whether to forward the message to the other handlers in the message chain. If you are performing your own spooling and overriding all of the spooling and despooling messages, you don't need to forward the messages. However, if you are not handling all of the spooling yourself, you do need to forward each message so that the default implementation can perform its task.

GXCreateSpoolFile

QuickDraw GX sends the `GXCreateSpoolFile` message at the start of spooling a document. You can override the `GXCreateSpoolFile` message to perform the setup for spool files. Your override of the `GXCreateSpoolFile` message must match the following formal declaration:

```
OSErr MyCreateSpoolFile (FSSpecPtr aFSSpecPtr,
                         long createOptions, gxSpoolFile *aSpoolFile);
```

`aFSSpecPtr` A pointer to a file specification that indicates where the spool file is created.

`createOptions`
Options for creating the spool file, as shown in Table 4-5.

`aSpoolFile`
A pointer to a file specification, which on return is a value that references the newly created spool file.

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

DESCRIPTION

QuickDraw GX sends the `GXCreateSpoolFile` message at the start of spooling, when an application calls the `GXStartJob` function to create a new spool file. Several options for creating the file may be specified by QuickDraw GX, as shown in Table 4-5.

Table 4-5 Options for the `GXCreateSpoolFile` message

Constant	Value	Explanation
`gxNoCreateOptions`	`0x00000000`	No options for this file: just create it.
`gxInhibitAlias`	`0x00000001`	Do not create an alias for this file in the Print Monitor Documents (PMD) folder.
`gxInhibitUniqueName`	`0x00000002`	Do not append anything to the filename to make it unique. If you specify this option and the file already exists, the file is replaced.
`gxResolveBitmapAlias`	`0x00000004`	Include bitmap data instead of the alias for the bitmap data.

If you override the `GXCreateSpoolFile` message, you can perform any setup you wish. You can override this message to change the location of the spool file by changing the contents of the `aFSSpecPtr` parameter. You can also override this message and all subsequent spooling messages if you want to do all your own spooling. The code in your override of this message is executed once per spooled job.

SPECIAL CONSIDERATIONS

You rarely send the `GXCreateSpoolFile` message yourself.

If you are providing your own spooling, you need to totally override the `GXCreateSpoolFile` message and all of the other spooling and despooling messages.

If you are not providing your own spooling (which is almost always the case), you must forward the `GXCreateSpoolFile` message because QuickDraw GX's default implementation creates a new spool file that is used by the other spooling messages.

RESULT CODES

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

SEE ALSO

The `GXStartJob` function is described in *Inside Macintosh: QuickDraw GX Printing*.

GXSpoolPage

QuickDraw GX sends the `GXSpoolPage` message when a page is ready to be sent to the spool file. You can override the `GXSpoolPage` message to change or add to a page being sent to a spool file. Your override of the `GXSpoolPage` message must match the following formal declaration:

```
OSErr MySpoolPage (gxSpoolFile aSpoolFile, gxFormat aFormat,
                   gxShape aShape);
```

`aSpoolFile`	The file to which the page is being written.
`aFormat`	The format that goes with the page.
`aShape`	The data that belongs on the page in the form of a picture shape.

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

DESCRIPTION

QuickDraw GX sends the `GXSpoolPage` message when an application calls either the `GXFinishPage` function or the `GXPrintPage` function. This message takes a picture shape in the `aShape` parameter that represents a page and writes it to the spool file referenced by the `aSpoolFile` parameter.

You override this message when you need to perform any per-page operations such as adding data (for example, a background picture or confidential stamp) to each page. You need to change the page before forwarding this message because the page is written to the file before control returns from the forwarded message.

The default implementation of `GXSpoolPage` writes the data into the spool file in a standard format.

SPECIAL CONSIDERATIONS

You rarely send the `GXSpoolPage` message yourself.

If you are providing your own spooling, you need to totally override `GXSpoolPage` and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward this message to allow the default implementation to write the data into the spool file.

RESULT CODES

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

SEE ALSO

The `GXFinishPage` and `GXPrintPage` functions are described in *Inside Macintosh: QuickDraw GX Printing*.

GXSpoolData

QuickDraw GX sends the `GXSpoolData` message whenever a stream of data is about to be written to the spool file. You can override the `GXSpoolData` message to modify data going into the spool file or to spool data in your own way. Your override of the `GXSpoolData` message must match the following formal declaration:

```
OSErr MySpoolData (gxSpoolFile aSpoolFile, Ptr data,
                   long *length);
```

`aSpoolFile`
The spool file to which the data is written.

`data`	A pointer to the buffer containing the data.
`length`	On entry, the length of the buffer in bytes. On return, the actual number of bytes that were spooled.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXSpoolData` message during spooling when a stream of data is about to be written to the spool file.

One reason to override this message is to encrypt the data in the spool file. If you do change the data in the spool file with an override of the `GXSpoolData` message, you must do so prior to forwarding the message.

The default implementation of the `GXSpoolData` message stores the data into the spool file. In most cases, you forward the `GXSpoolData` message after making your data changes and let the default implementation write the data bytes.

SPECIAL CONSIDERATIONS

You rarely send the `GXSpoolData` message yourself.

If you are providing your own spooling, you need to totally override the `GXSpoolData` message and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward the `GXSpoolData` message to allow the default implementation to write the data into the spool file.

RESULT CODES

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

GXSpoolResource

QuickDraw GX sends the `GXSpoolResource` message when a resource is about to be added to the spool file. You can override the `GXSpoolResource` message to add your own resource to the spool file. Your override of the `GXSpoolResource` message must match the following formal declaration:

```
OSErr MySpoolResource (gxSpoolFile aSpoolFile, Handle aResource,
                        ResType aType, short id);
```

`aSpoolFile`	The spool file to which you are adding resources.

`aResource`	A handle to the resource that you want added to the spool file.
`aType`	The resource type of the resource that you are adding.
`id`	The resource ID of the resource that you are adding.

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

DESCRIPTION

QuickDraw GX sends the `GXSpoolResource` message after a message handler has issued a `GXSpoolResource` message to write a resource to the spool file.

You can override this message if you need to change the data in a resource that is being added to the spool file. For example, if you are encrypting the data in a spool file, you might want to override this message and encrypt the resource data before it is written to the file.

The default implementation of `GXSpoolResource` expects the `aResource` handle to be a normal memory handle, not a resource. After this message executes, the handle has become a resource handle that the caller cannot use. This behavior is the same as for the `AddResource` function, which is described in the chapter “Resource Manager” in *Inside Macintosh: More Macintosh Toolbox*.

SPECIAL CONSIDERATIONS

You can send the `GXSpoolResource` message yourself if you have a resource that you want to store in the spool file.

If you are providing your own spooling, you need to totally override the `GXSpoolResource` message and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward the `GXSpoolResource` message to allow the default implementation to write the data into the spool file.

RESULT CODES

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

GXCompleteSpoolFile

QuickDraw GX sends the `GXCompleteSpoolFile` message when spooling of a document has been completed and the spool file is about to be closed. You can override the `GXCompleteSpoolFile` message to conclude operations on a spool file. Your override of the `GXCompleteSpoolFile` message must match the following formal declaration:

```
OSErr MyCompleteSpoolFile (gxSpoolFile aSpoolFile);
```

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

DESCRIPTION

QuickDraw GX, in its default implementation of the `GXFinishJob` message, sends the `GXCompleteSpoolFile` message when spooling is complete and the spool file is about to be closed.

You can override this message to determine when a spool file is closing or to conclude your operations on the spool file. You can also override this message if you have code you wish to execute once per spooled job.

The default implementation of the `GXCompleteSpoolFile` message finishes the spooling process and closes the file.

SPECIAL CONSIDERATIONS

You rarely send the `GXCompleteSpoolFile` message yourself.

If you are providing your own spooling, you need to totally override the `GXCompleteSpoolFile` message and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward the `GXCompleteSpoolFile` message to allow the default implementation to finish spooling data and close the spool file.

RESULT CODES

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

SEE ALSO

The `GXFinishJob` message is described on page 4-54.

Despooling Messages

Despooling messages are sent during the imaging phase of printing to read the data from the disk and prepare it for rendering into a format that is acceptable for the printing device. You can override these messages to modify or change data just prior to it being rendered.

GXCountPages

QuickDraw GX sends the `GXCountPages` message to determine how many pages are in the file that is about to be despooled. You can override the `GXCountPages` message to change the logical number of pages in the spool file. Your override of the `GXCountPages` message must match the following formal declaration:

```
OSErr MyCountPages (gxSpoolFile aSpoolFile, long *numPages);
```

`aSpoolFile`	The spool file.
`numPages`	On return, the number of pages in the spool file.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX, in its default implementation of the `GXImageDocument` message, sends the `GXCountPages` message to determine the number of pages in the spool file prior to despooling.

You override this message if you need to change the logical number of pages in the spool file (for example, if you are producing thumbnail sketches of the document page with several to a printed page).

The default implementation of this message counts and returns the number of pages in the spool file. If you need to alter the page count returned by this message, you need to first forward the message and then add your own code, as you would in the case of the thumbnail example.

SPECIAL CONSIDERATIONS

You rarely send the `GXCountPages` message yourself.

You must forward the `GXCountPages` message because QuickDraw GX depends on receiving it to know how many pages to despool. You need to first forward this message and then change the value, if necessary.

RESULT CODES

`gxSegmentLoadFailedErr`	A required code segment could not be found, or there was not enough memory to load it.
`gxPrUserAbortErr`	The user has canceled printing.
`gxIncompletePrintFileErr`	The spool file is not complete.
`gxCrashedPrintFileErr`	The spool file could not be opened.
`gxInvalidPrintFileVersion`	The version number of the spool file is not valid.

SEE ALSO

The `GXImageDocument` message is described on page 4-93.

GXDespoolPage

QuickDraw GX sends the `GXDespoolPage` message to obtain the next page to be printed from the spool file. If you perform your own spooling, you need to override the `GXDespoolPage` message to interpret your spool file format. Your override of the `GXDespoolPage` message must match the following formal declaration:

```
OSErr MyDespoolPage (gxSpoolFile aSpoolFile,
                     long pageNum, gxFormat aFormat,
                     gxShape *aShape, Boolean *formatChanged);
```

`aSpoolFile`	The spool file.
`pageNum`	The page number of the despooled page.
`aFormat`	A format object that is filled in by this function. This object needs to be preallocated.
`aShape`	On return, a reference to the data that belongs on the page in the form of a picture shape.
`formatChanged`	On return, a Boolean value that is `true` if the page contains a format that is different from the previous format returned by the `GXDespoolPage` message and `false` if not.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX, in its default implementation of the `GXImagePage` message, sends the `GXDespoolPage` message to obtain the next page in the spool file to be printed.

You can override this message to change the default settings in a newly created format object. If you override the `GXSpoolPage` message to encrypt data, you need to override `GXDespoolPage` to decrypt the data.

The default implementation of the `GXDespoolPage` message reads the format and shape data for this page. It reads the page data from the spool file and repeatedly sends the `GXDespoolData` message.

SPECIAL CONSIDERATIONS

You rarely send the `GXDespoolPage` message yourself.

If you are providing your own spooling, you need to totally override the `GXDespoolPage` message and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward the `GXDespoolPage` message to allow the default implementation to read the data from the spool file. Forward this message prior to modifying the graphics shape or related format object.

RESULT CODES

`gxSegmentLoadFailedErr`	A required code segment could not be found, or there was not enough memory to load it.
`gxPrUserAbortErr`	The user has canceled printing.
`gxIncompletePrintFileErr`	The spool file is not complete.
`gxCrashedPrintFileErr`	The spool file could not be opened.
`gxInvalidPrintFileVersion`	The version number of the spool file is not valid.

SEE ALSO

You can find an example of an overide of the `GXDespoolPage` message in Listing 2-14 on page 2-25.

The `GXImagePage` message is described on page 4-94.

The `GXSpoolPage` message is described on page 4-69.

The `GXDespoolData` message is described in the next section.

GXDespoolData

QuickDraw GX sends the `GXDespoolData` message to read a stream of data from the spool file. If you perform your own spooling, you need to override the `GXDespoolData` message to interpret your spool file format. Your override of the `GXDespoolData` message must match the following formal declaration:

```
OSErr MyDespoolData (gxSpoolFile aSpoolFile,
                     Ptr data, long *length);
```

`aSpoolFile`	The spool file.
`data`	A pointer to the buffer that holds the data from the spool file.
`length`	On entry, the length of the buffer in bytes. On return, the actual number of bytes that were despooled.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

You can override the `GXDespoolData` message to decrpyt data if you encrypted data in an override of the `GXSpoolData` message.

The default implementation of this message reads the requested amount of data from the spool file.

SPECIAL CONSIDERATIONS

You rarely send the `GXDespoolData` message yourself.

If you are providing your own spooling, you need to totally override the `GXDespoolData` message and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward the `GXDespoolData` message to allow the default implementation to read the data from the spool file. Forward this message prior to modifying the data.

RESULT CODES

`gxSegmentLoadFailedErr`	A required code segment could not be found, or there was not enough memory to load it.
`gxPrUserAbortErr`	The user has canceled printing.
`gxIncompletePrintFileErr`	The spool file is not complete.
`gxCrashedPrintFileErr`	The spool file could not be opened.
`gxInvalidPrintFileVersion`	The version number of the spool file is not valid.

SEE ALSO

The `GXSpoolData` message is described on page 4-70.

GXDespoolResource

QuickDraw GX sends the `GXDespoolResource` message to read a resource from the spool file. If you perform your own spooling, you need to override the

`GXDespoolResource` message to interpret your spool file format. Your override of the `GXDespoolResource` message must match the following formal declaration:

```
OSErr MyDespoolResource (gxSpoolFile aSpoolFile,
                         ResType aType, short id,
                         Handle *aResource);
```

`aSpoolFile`	The spool file.
`aType`	The resource type of the resource to read.
`id`	The resource ID of the resource to read.
`aResource`	On return, a handle to the resource that was despooled.

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

DESCRIPTION

You can override the `GXDespoolResource` message to decrypt a resource if you encrypted a resource in an override of the `GXSpoolResource` message.

The default implementation of this message reads the requested resource from the spool file.

SPECIAL CONSIDERATIONS

You rarely send the `GXDespoolResource` message yourself.

If you are providing your own spooling, you need to totally override the `GXDespoolResource` message and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward the `GXDespoolResource` message to allow the default implementation to read the resource data from the spool file. Forward this message prior to modifying the resource data.

RESULT CODES

`gxSegmentLoadFailedErr`	A required code segment could not be found, or there was not enough memory to load it.
`gxPrUserAbortErr`	The user has canceled printing.
`gxIncompletePrintFileErr`	The spool file is not complete.
`gxCrashedPrintFileErr`	The spool file could not be opened.
`gxInvalidPrintFileVersion`	The version number of the spool file is not valid.

SEE ALSO

The `GXSpoolResource` is described on page 4-71.

GXExamineSpoolFile

QuickDraw GX sends the `GXExamineSpoolFile` message just prior to establishing communications with a device. You need to override the `GXExamineSpoolFile` message if you have your own spool file format. Your override code of the `GXExamineSpoolFile` message must match the following formal declaration:

```
OSErr MyExamineSpoolFile (gxSpoolFile aSpoolFile);
```

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

DESCRIPTION

QuickDraw GX sends the `GXExamineSpoolFile` message just prior to opening the connection to the device.

If you are using the default spool-file format, you rarely need to override the `GXExamineSpoolFile` message.

The default implementation of this message checks the spool file for consistency before allowing it to be printed.

SPECIAL CONSIDERATIONS

You never send the `GXExamineSpoolFile` message yourself. You rarely need to override this message.

If you are using your own spool file format, you need to totally override the `GXExamineSpoolFile` message. Otherwise, you need to forward it to other handlers, either before or after performing your own tasks.

RESULT CODES

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

GXCloseSpoolFile

QuickDraw GX sends the `GXCloseSpoolFile` message to close the spool file. If you perform your own spooling, you need to override the `GXCloseSpoolFile` message to handle closing the spool file and updating it on disk. Your override of the `GXCloseSpoolFile` message must match the following formal declaration:

```
OSErr MyCloseSpoolFile (gxSpoolFile aSpoolFile,
                        long closeOptions);
```

`aSpoolFile`
The spool file.

`closeOptions`
Options for closing the spool file, as shown in Table 4-6.

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

DESCRIPTION

QuickDraw GX sends the `GXCloseSpoolFile` message to close and possibly delete a spool file once imaging is complete. You can specify various options in the `closeOptions` parameter. The constants for the spool-file closing options are shown in Table 4-6.

Table 4-6 `GXCloseSpoolFile` options

Constant	Value	Explanation
`gxNoCloseOptions`	`0x00000000`	No options are in effect
`gxDeleteOnClose`	`0x00000001`	The spool file is to be deleted rather than closed
`gxUpdateJobData`	`0x00000002`	QuickDraw GX needs to write the current job information into the spool file before closing it
`gxMakeRemoteFile`	`0x00000004`	QuickDraw GX needs to set the type of the spool file to `'rjob'` rather than deleting it, which means that the file is sent to another machine for printing and that the original file is retained in case an error occurs

You can override the `GXCloseSpoolFile` message to perform any operations that you need to when the spool file is closed.

The default implementation of the `GXCloseSpoolFile` message includes the `gxDeleteOnClose` option.

SPECIAL CONSIDERATIONS

You rarely send the `GXCloseSpoolFile` message yourself.

If you are providing your own spooling, you need to totally override the `GXCloseSpoolFile` message and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward the `GXCloseSpoolFile` message to allow the default implementation to close the spool file.

RESULT CODES

`gxSegmentLoadFailedErr`	A required code segment could not be found, or there was not enough memory to load it.
`gxPrUserAbortErr`	The user has canceled printing.
`gxIncompletePrintFileErr`	The spool file is not complete.
`gxCrashedPrintFileErr`	The spool file could not be opened.
`gxInvalidPrintFileVersion`	The version number of the spool file is not valid.

Dialog Box Messages

You can use dialog box messages to communicate with users through dialog boxes. QuickDraw GX sends dialog box messages during the application phase of printing when the user interacts with the print dialog boxes. You can override these messages to add panels or modify the panels you present to the user.

GXPrintingEvent

QuickDraw GX sends the `GXPrintingEvent` message when events occur in a print dialog box. You can override the `GXPrintingEvent` message to handle events such as window update events that occur during display of print dialog boxes. Your override of the `GXPrintingEvent` message must match the following formal declaration:

```
OSErr MyPrintingEvent (EventRecord *anEventRecord,
                       Boolean filterEvent);
```

`anEventRecord`
A pointer to an event that occurred in a print dialog box.

`filterEvent`
A Boolean value that is `true` if the event needs to be filtered, and `false` if not.

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

DESCRIPTION

QuickDraw GX sends the `GXPrintingEvent` message whenever a specific event occurs in one of the print dialog boxes that is displayed for printing. You can override this message if you need to process events that occur during printing.

The default implementation of this message does nothing. Application programs must override this message to correctly support print dialog boxes.

SPECIAL CONSIDERATIONS

You never send the `GXPrintingEvent` message yourself.

You always create a total override of the `GXPrintingEvent` message.

RESULT CODES

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

GXJobDefaultFormatDialog

QuickDraw GX sends the `GXJobDefaultFormatDialog` message when the application displays the Page Setup dialog box. You can override the `GXJobDefaultFormatDialog` message to modify the behavior or appearance of the dialog box. Your override of the `GXJobDefaultFormatDialog` message must match the following formal declaration:

```
OSErr MyJobDefaultFormatDialog (gxDialogResult *aDialogResult);
```

`aDialogResult`
On return, a poiner to a value that specifies the selection made by the user in the dialog box.

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

DESCRIPTION

QuickDraw GX sends the `GXJobDefaultFormatDialog` message when the user clicks the More Choices button in the Page Setup dialog box. The application calls the `GXJobDefaultFormatDialog` function to display the extended Page Setup dialog box.

The default implementation of this message adds the standard printing panels and interface and then displays the dialog box.

You usually override this message to customize the dialog box by adding panels using the `GXSetupDialogPanel` function. You can add your own panels to the dialog box through the normal QuickDraw GX printing calls.

SPECIAL CONSIDERATIONS

You never send the `GXJobDefaultFormatDialog` message yourself.

You must forward the `GXJobDefaultFormatDialog` message to other message handlers. Add your panels and then forward the message.

RESULT CODES

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

SEE ALSO

You can find an example of an override of the `GXJobDefaultFormatDialog` message in Listing 3-7 on page 3-29 in the chapter "Printer Drivers."

The `GXSetupDialogPanel` function is described on page 5-28 in the chapter "Printing Functions for Message Overrides."

The `GXJobDefaultFormatDialog` function is described in *Inside Macintosh: QuickDraw GX Printing*.

GXFormatDialog

QuickDraw GX sends the `GXFormatDialog` message when the application displays the Custom Page Setup dialog box. You can override the `GXFormatDialog` message to modify the behavior or appearance of the dialog box. Your override of the `GXFormatDialog` message must match the following formal declaration:

```
OSErr MyFormatDialog (gxFormat aFormat, StringPtr title,
                      gxDialogResult, *aDialogResult);
```

`aFormat`	A reference to the format object.
`title`	The title of the dialog box. If you specify `nil` as the value of this parameter, the title "Custom Page Setup" is used.
`aDialogResult`	On return, a pointer to the selection made by the user in the dialog box.

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

DESCRIPTION

QuickDraw GX sends the `GXFormatDialog` message when the user selects the Custom Page Setup menu item and an application subsequently calls the `GXFormatDialog` function to display the Custom Page Setup dialog box.

The default implementation of this message adds the standard printing panels and interface and then displays the dialog box.

You usually override this message to customize the dialog box by adding panels using the `GXSetupDialogPanel` function.

SPECIAL CONSIDERATIONS

You never send the `GXFormatDialog` message yourself.

You must forward the `GXFormatDialog` message to other message handlers. Add your panels and then forward the message.

RESULT CODES

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

SEE ALSO

The `GXFormatDialog` function is described in *Inside Macintosh: QuickDraw GX Printing*.

The `GXSetupDialogPanel` function is described on page 5-28 in the chapter "Printing Functions for Message Overrides."

GXJobPrintDialog

QuickDraw GX sends the `GXJobPrintDialog` message when the application displays the Print dialog box. You can override the `GXJobPrintDialog` message to modify the behavior or appearance of the Print dialog box. Your override of the `GXJobPrintDialog` message must match the following formal declaration:

```
OSErr MyJobPrintDialog (gxDialogResult *aDialogResult);
```

`aDialogResult`
On return, a pointer to the selection made by the user in the dialog box.

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

DESCRIPTION

QuickDraw GX sends the `GXJobPrintDialog` message when the user selects Print from the File menu and the application subsequently calls the `GXJobPrintDialog` function to display the Print dialog box on the user's screen.

The default implementation of this message adds the standard printing panels and interface and then displays the dialog box.

You usually override this message to customize the dialog box by adding panels using the `GXSetupDialogPanel` function.

SPECIAL CONSIDERATIONS

You never send the `GXJobPrintDialog` message yourself.

You must forward the `GXJobPrintDialog` message to other message handlers. Add your panels and then forward the message.

RESULT CODES

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

SEE ALSO

You can find an example of an override of the `GXJobPrintDialog` message in Listing 2-3 on page 2-13 in the chapter "Printing Extensions."

The `GXSetupDialogPanel` function is described in the section "Adding a Panel to a Print Dialog Box" beginning on page 5-27 in the chapter "Printing Functions for Message Overrides."

GXHandlePanelEvent

QuickDraw GX sends the `GXHandlePanelEvent` message when an event happens in a panel. You can override the `GXHandlePanelEvent` message to handle panel events that cannot be handled using `'xdtl'` resources. Your override of the `GXHandlePanelEvent` message must match the following formal declaration:

```
OSErr MyHandlePanelEvent (gxPanelInfoRecord *aPanelInfoRecord,
                          gxPanelResult *panelResult);
```

`aPanelInfoRecord`
: A pointer to the panel information structure that supplies information to the panel about the current dialog box and panel event.

`panelResult`
: On return, the result of handling the panel event. This is one of the values described in the section "Panel Responses" on page 4-37.

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

DESCRIPTION

QuickDraw GX sends the `GXHandlePanelEvent` message to allow a panel to handle events associated with the dialog box.

The default implementation of this message does nothing. You need to override this message if you add panels that cannot be handled in a standard way (using `'xdtl'` resources).

SPECIAL CONSIDERATIONS

You never send the `GXHandlePanelEvent` message yourself.

You always perform a total override of the `GXHandlePanelEvent` message, in which you handle any events of interest that occur in your panel.

RESULT CODES

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

SEE ALSO

The panel information structure is described on page 4-35.

You can find an example of an overide of the `GXHandlePanelEvent` message in Listing 2-12 on page 2-22 in the chapter "Printing Extensions."

Print dialog boxes, panels, and the `'xdtl'` resource are described in *Inside Macintosh: QuickDraw GX Printing*.

GXFilterPanelEvent

QuickDraw GX sends the `GXFilterPanelEvent` message when an event happens in a panel. You can override the `GXFilterPanelEvent` message to add panels that need a filter procedure. Your override of the `GXFilterPanelEvent` message must match the following formal declaration:

```
OSErr MyFilterPanelEvent (gxPanelInfoRecord *aPanelInfoRecord;
                          Boolean *returnImmed);
```

`aPanelInfoRecord`
: A pointer to the panel information structure that supplies information to the panel about the current dialog box and panel event.

`returnImmed`
: On return, a Boolean value that is `true` if there should be no further processing on this event and `false` if not.

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

DESCRIPTION

QuickDraw GX sends the `GXFilterPanelEvent` message to filter panel events in a dialog box.

The default implementation of this message does nothing. You need to override this message if you add panels that require a filtering process.

SPECIAL CONSIDERATIONS

You never send the `GXFilterPanelEvent` message yourself.

You always perform a total override of the `GXFilterPanelEvent` message, in which you filter any events that occur in your panel.

RESULT CODES

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

SEE ALSO

The panel information structure is described on page 4-35.

Print dialog boxes and panels are described in *Inside Macintosh: QuickDraw GX Printing.*

Universal Imaging Messages

QuickDraw GX sends universal imaging messages during the imaging phase of printing to allow you to change a spool file into a format more readily accepted by the printer. In addition to these messages, you will see messages that are specific to each type of printer that the system supports: raster, PostScript, and vector.

GXJobStatus

QuickDraw GX sends the `GXJobStatus` message to display the current status of a print job during spooling and despooling. You can override the `GXJobStatus` message to handle status at spooling and despooling times. Your override of the `GXJobStatus` message must match the following formal declaration:

```
OSErr MyJobStatus (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.

An example of why you might override this message is to display the status in a window.

The default implementation of this message displays the status in the desktop printer window.

SPECIAL CONSIDERATIONS

You never send the `GXJobStatus` message yourself.

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

RESULT CODES

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

SEE ALSO

The status structure is described in the section “The Status Structure” beginning on page 4-39.

The `GXReportStatus` function is described on page 5-17 in the chapter “Printing Functions for Message Overrides.”

GXCaptureOutputDevice

QuickDraw GX sends the `GXCaptureOutputDevice` message to remove a device from a network or to reestablish its availability on the network. If you write drivers that communicate with network devices (other than PAP) that can be removed from the network, you can override the `GXCaptureOutputDevice` message to manage the capture and release of these devices. Your override of the `GXCaptureOutputDevice` message must match the following formal declaration:

```
OSErr MyCaptureOutputDevice (Boolean capture);
```

`capture`	A Boolean value that is `true` if you want the device captured and `false` if you want it released.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXCaptureOutputDevice` message when it needs to remove a printing device from the network or to return the printing device to availability on the network.

The default implementation of the `GXCaptureOutputDevice` message automatically handles the capturing and restoring of PAP devices on a network. It uses a series of capture resources to control the capture process.

You can override this message if you write drivers that implement communications with network devices (other than PAP) that can be removed from the network. Your override implements the capture and release of these devices.

SPECIAL CONSIDERATIONS

You never send the `GXCaptureOutputDevice` message yourself.

You always implement a total override of the `GXCaptureOutputDevice` message, in which you implement your own device capture and release strategy.

RESULT CODES

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

SEE ALSO

Capture resources are described in the section “The Capture ('cpts') Resource” beginning on page 6-63 in the chapter “Printing Resources.”

GXImageJob

QuickDraw GX sends the `GXImageJob` message at the start of the imaging phase of printing, when a print job is ready to print. You can override the `GXImageJob` message to affect the way an entire job is printed. Your override of the `GXImageJob` message must match the following formal declaration:

```
OSErr MyImageJob (gxSpoolFile aSpoolFile, long *closeOptions);
```

`aSpoolFile`	The spool file.
`closeOptions`	A pointer to the spool-file closing options for the job. These are passed on to the `GXCloseSpoolFile` message at the end of spooling. You can use the values that are shown in Table 4-6 on page 4-80.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXImageJob` message at the start of imaging when it determines that a print job in the queue is ready to be printed.

The default implementation of `GXImageJob` prints an entire spool file. It first determines whether the file is to be printed on the local Macintosh or sent to a remote station. In the default implementation, QuickDraw GX sends the `GXOpenConnection` message to open the printer connection. Next, it sends the `GXSetupImageData` message so that the imaging system can set up its imaging time data. It also sends the `GXImageDocument` message to begin imaging the document. And finally, it sends the `GXCloseConnection` message to close the connection to the printer.

SPECIAL CONSIDERATIONS

You never send the `GXImageJob` message yourself.

You must forward the `GXImageJob` message to other message handlers. You can perform your tasks before or after forwarding the message.

RESULT CODES

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

SEE ALSO

The `GXCloseSpoolFile` message is described on page 4-79.

The `GXSetupImageData` message is described on page 4-92.

The `GXImageDocument` message is described on page 4-93.

The `GXOpenConnection` message is described on page 4-131.

The `GXCloseCOnnection` message is described on page 4-135.

GXCreateImageFile

QuickDraw GX sends the `GXCreateImageFile` message before the imaging of a print job begins. You need to override the `GXCreateImageFile` message if your device requires that an image file be created. Your override of the `GXCreateImageFile` message must match the following formal declaration:

```
OSErr MyCreateImageFile (FSSpecPtr aFSSpecPtr,
                         long imageFileOptions,
                         long *fileReference);
```

`aFSSpecPtr`	Where to create the image file.
`imageFileOptions`	Options for creation of the image file, as shown in Table 4-7.
`fileReference`	On return, this value specifies a reference to the created image file.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

You need to override the `GXCreateImageFile` message if you are working with an output device that needs to be driven from a file. For example, a film recorder might need to process all of the data for a single image plane at once, due to hardware timing constraints. Or a facsimile machine might need to receive the data for a page without lengthy interruptions that indicate a disconnection. For devices with constraints like these, you need to send the printer data to a file that is subsequently streamed to the device.

When you create an imaging file, QuickDraw GX stores the printing data and plays it back to the output device in a steady stream. The options that you specify in the `imageFileOptions` parameter define how the playback occurs, as shown in Table 4-7.

The default implementation of this message creates the image file if the options specify that it should.

Table 4-7 Image file options

Constant	Value	Explanation
`gxNoImageFile`	0	The function does not create an image file. This is the default value.
`gxMakeImageFile`	1	The function creates an image file.
`gxEachPlane`	2	The function stores one image plane of data at a time. This allows a device like a film recorder to process each image plane of a data without pausing.
`gxEachPage`	4	The function stores one page of data at a time. This allows a device to process each page without pausing.
`gxEntireFile`	6	The function stores the entire document to allow the device to process the document without pausing.

SPECIAL CONSIDERATIONS

You never send the `GXCreateImageFile` message yourself.

You must forward the `GXCreateImageFile` message to other message handlers. You can modify the options and then forward the message.

RESULT CODES

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

GXSetupImageData

QuickDraw GX sends the `GXSetupImageData` message to allow you to send initialization data that is specific to the kind of imaging system (raster, PostScript, or vector) that is being used for a job. You need to override the `GXSetupImageData` message if you want to modify imaging data. Your override of the `GXSetupImageData` message must match the following formal declaration:

```
OSErr MySetupImageData (void *imageData);
```

`imageData`	A pointer to imaging-system-specific data for initializing the printing device.

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

DESCRIPTION

QuickDraw GX sends the `GXSetupImageData` message to initialize data specific to the type of printer that is being printed to: raster, PostScript, or vector.

The default implementation of this message reads the default imaging information from resources within the specific driver.

SPECIAL CONSIDERATIONS

You never send the `GXSetupImageData` message yourself.

You almost always forward the `GXSetupImageData` message to other message handlers. Forward the message to get the default values of the imaging data filled in, and then change the data as you require.

RESULT CODES

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

SEE ALSO

You can find an example of an override of the `GXSetupImageData` message in Listing 3-11 on page 3-36 in the chapter "Printer Drivers."

The raster imaging system structure, which contains the data used by the raster imaging system, is described on page 4-23.

The PostScript imaging system structure, which contains the data used by the PostScript imaging system, is described on page 4-26.

The vector imaging system structure, which contains the data used by the vector imaging system, is described on page 4-32.

GXImageDocument

QuickDraw GX sends the `GXImageDocument` message just prior to starting the imaging of a document. You need to override the `GXImageDocument` message if you want to perform a task at the start of imaging for a document. Your override of the `GXImageDocument` message must match the following formal declaration:

```
OSErr MyImageDocument (gxSpoolFile aSpoolFile, void *imageData);
```

`aSpoolFile`	The spool file to image.
`imageData`	A pointer to imaging-system-specific data.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXImageDocument` message to print a spool file.

The default implementation of the `GXImageDocument` message prints the document. First, it creates a new format to use when it calls the `GXDespoolPage` message. Then it sends the `GXCountPages` message to find out how many pages are on a spool file. It loops, sending the `GXImagePage` message for each page in the document. And finally, it disposes of the format that it allocated.

SPECIAL CONSIDERATIONS

You never send the `GXImageDocument` message yourself.

You almost always forward the `GXImageDocument` message to other message handlers. You can forward the message before or after performing your own tasks.

RESULT CODES

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

SEE ALSO

The `GXDespoolPage` message is described on page 4-75.

The `GXCountPages` message is described on page 4-74.

The `GXImagePage` message is described on page 4-94.

GXImagePage

QuickDraw GX sends the `GXImagePage` message just prior to starting the imaging of a page. You need to override the `GXImagePage` message if you want to perform some task for the imaging of each page. Your override of the `GXImagePage` message must match the following formal declaration:

```
OSErr MyImagePage (gxSpoolFile aSpoolFile, long pageNumber,
                   gxFormat aFormat, void *imageData);
```

`aSpoolFile`	The file to print.
`pageNumber`	The page being imaged.
`aFormat`	The format object for the page.
`imageData`	A pointer to imaging-system-specific data.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX, in its default implementation of the `GXImageDocument` message, sends the `GXImagePage` message once for each page in a spool file that is included in the count returned by the `GXCountPages` message.

The default implementation of this message prints the page. First, it sends the `GXStartSendPage` message to indicate that a page is being sent to a printer. Next, it sends the `GXRenderPage` message so that the imaging system can translate the page from graphics data into data for the specified printer. It then sends the `GXFinishSendPage` message to indicate that the page has been sent to the printer.

SPECIAL CONSIDERATIONS

You never send the `GXImagePage` message yourself.

You almost always forward the `GXImagePage` message to other message handlers. You can forward the message before or after performing your own tasks.

RESULT CODES

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

SEE ALSO

The `GXImageDocument` message is described in the previous section.

The `GXCountPages` message is described on page 4-74.

The `GXStartSendPage` message is described on page 4-136.

The `GXRenderPage` message is described on page 4-96.

The `GXFinishSendPage` message is described on page 4-138.

GXRenderPage

QuickDraw GX sends the `GXRenderPage` message just prior to the rendering the image of a page. You need to override the `GXRenderPage` message if you want to perform some task during rendering of each page. Your override of the `GXRenderPage` message must match the following formal declaration:

```
OSErr MyRenderPage (gxFormat aFormat, gxShape aShape,
                     gxPageInfoRecord *aPageInfoRecord,
                     void *imageData);
```

`aFormat`	The format object for the page.
`aShape`	The data that belongs on the page in the form of a graphics picture shape.
`aPageInfoRecord`	A pointer to a page information structure.
`imageData`	A pointer to imaging-system-specific data.

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

DESCRIPTION

The default implementation of the `GXImagePage` message sends the `GXRenderPage` message once for each copy of a page. The `GXRenderPage` message renders a single page into raster, PostScript, or vector format.

You need to override this message if you want to perform some task prior to, during, or after the rendering of each page.

The default implementation of the `GXRenderPage` message renders one copy of the page, sending messages to relay that information to the specific driver.

SPECIAL CONSIDERATIONS

You never send the `GXRenderPage` message yourself.

You almost always forward the `GXRenderPage` message to other message handlers. You can forward the message before or after performing your own tasks.

RESULT CODES

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

SEE ALSO

The page information structure is described on page 4-11.

The `GXImagePage` message is described in the previous section.

Raster Imaging Messages

Raster imaging messages are sent only when the specific driver is imaging for a raster-based device. The type of device is selected when you specify raster imaging in the imaging system resource for your printing extension or printer driver. The imaging system (`'isys'`) resource is described on page 6-33 in the chapter "Printing Resources."

GXRasterDataIn

You can override the `GXRasterDataIn` message to send data to a raster device for printing. Your override of the `GXRasterDataIn` message must match the following formal declaration:

```
OSErr MyRasterDataIn (gxOffscreenHdl offScreen,
         gxRectangle *bandRectangle, gxRectangle *dirtyRectangle);
```

`offScreen` Data representing this portion of the page.

`bandRectangle`
A pointer to a rectangle that defines the size and location of the band of data that is being passed in.

`dirtyRectangle`
A pointer to a rectangle that defines the area of `bandRectangle` that is occupied by shapes.

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

DESCRIPTION

QuickDraw GX, in its default implementation of the `GXRenderPage` message for raster drivers, sends the `GXRasterDataIn` message after rendering a single band of data. The raster driver sends page data to the device in response to this message. Since a page cannot always be rendered as a single bitmap, QuickDraw GX can send this message multiple times for a page.

You can override this message to convert the bitmaps sent by QuickDraw GX into a format that is acceptable for your raster device. You can call the `GXBufferData` or `GXWriteData` messages to send the data to the device.

The default implementation of the `GXRasterDataIn` message breaks the data up into smaller pieces by subdividing the bitmap into "head passes" as defined in the raster package resource. It then sends `GXRasterLineFeed` and `GXRasterPackageBitmap` messages to further break down the data and prepare it for the device.

SPECIAL CONSIDERATIONS

You never send the `GXRasterDataIn` message yourself.

You can totally override the `GXRasterDataIn` message, or you can modify the data in some way and then forward the message.

If you use the default implementation of the `GXRasterDataIn` message, you must define a raster package (`'rpck'`) resource.

RESULT CODES

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

SEE ALSO

The raster package resource is described on page 6-73 in the chapter "Printing Resources."

The `GXRenderPage` message is described on page 4-96.

The `GXRasterLineFeed` message is described on page 4-98.

The `GXRasterPackageBitmap` message is described on page 4-100.

The `GXBufferData` message is described on page 4-139.

The `GXWriteData` message is described on page 4-141.

GXRasterLineFeed

You can override the `GXRasterLineFeed` message to send the codes to a printing device that cause it to move the print head. Your override of the `GXRasterLineFeed` message must match the following formal declaration:

```
OSErr MyRasterLineFeed (short *lineFeedSize,
                        Ptr buffer, unsigned long *bufferPos,
                        gxRasterImageDataHdl imageData);
```

`lineFeedSize`	The amount by which to move the print head.
`buffer`	A pointer to the buffer for creating the line-feed sequence.
`bufferPos`	The location in the buffer to create the sequence.
`imageData`	A pointer to raster imaging-system-specific data. This structure is described in the section "Raster Imaging System Structure" beginning on page 4-23.

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

DESCRIPTION

QuickDraw GX, in its default implementation of the `GXRasterDataIn` message, sends the `GXRasterLineFeed` message to move the print head up or down the page. This message creates a sequence of characters for the device to move the print head up or down the page, by an amount specified in the `lineFeedSize` parameter. The sequence of characters needs to be placed in the buffer (specified in the `bufferPos` parameter) from the beginning of the pointer.

The default implementation of this message sends the line-feed in a format specified in the raster package control resource for your driver. It uses specific strings defined in that resource to control the process of generating the sequence of characters from the requested line-feed size.

You can override this message to provide your own translation from the line-feed size into the sequence of characters. You override of the `GXRasterLineFeed` message needs to translate the line-feed size into a sequence of characters, store this sequence in the buffer, decrement the line-feed size by the amount translated, and increment the location in the buffer by the number of bytes added to the buffer.

Note

The value of the line-feed size may be any integer value (negative, zero, or positive). u

SPECIAL CONSIDERATIONS

You never send the `GXRasterLineFeed` message yourself.

You can totally override the `GXRasterLineFeed` message, or you can modify the data in some way and then forward the message.

If you use the default implementation of the `GXRasterLineFeed` message, you must define a raster package control (`'ropt'`) resource.

RESULT CODES

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

SEE ALSO

You can find an example of an override of the `GXRasterLineFeed` message in Listing 3-3 on page 3-19 in the chapter “Printer Drivers.”

The raster package control resource is described on page 6-74 in the chapter “Printing Resources.”

The `GXRasterDataIn` message is described on page 4-97.

The raster imaging system structure is described on page 4-23.

GXRasterPackageBitmap

QuickDraw GX sends the `GXRasterPackageBitmap` message to translate a portion of a bitmap into a sequence of characters for an output device to print. If you are writing a printing extension or printer driver, you can override the `GXRasterPackageBitmap` message to perform your own translation from bitmaps into character sequences for your output device. Your override of the `GXRasterPackageBitmap` message must match the following formal declaration:

```
OSErr MyRasterPackageBitmap (
                        gxRasterPackageBitmapRec *whattoPackage,
                        Ptr buffer, unsigned long *bufferPos,
                        gxRasterImageDataHdl imageData);
```

`whattoPackage`	The bitmap to translate and which part of it to package.
`buffer`	The buffer for creating the sequence of characters.
`bufferPos`	The location in the buffer to begin placing characters.
`imageData`	A pointer to raster imaging-system-specific data. This structure is described in the section "Raster Imaging System Structure" beginning on page 4-23.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX, in its default implementation of the `GXRasterDataIn` message, sends the `GXRasterPackageBitmap` message to create a sequence of characters that represent a portion of the bitmap on the device. This message is sent multiple times per bitmap: once for each head pass that is required to print the bitmap, as specified in the raster package (`'rpck'`) resource.

The default implementation of this message does nothing.

You always perform a total override of this message in your raster device driver to provide your own translation from bitmaps into a sequence of characters for the device. After translating part of the bitmap into a sequence of characters and storing this sequence into the buffer, you increment the buffer position (specified in the `bufferPos` parameter) by the number of bytes added to the buffer.

SPECIAL CONSIDERATIONS

You never send the `GXRasterPackageBitmap` message yourself.

You can partially override the `GXRasterPackageBitmap` message in an extension. In a raster device driver, you always totally override this message.

RESULT CODES

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

SEE ALSO

The `gxRasterPackageBitmapRec` data type is described in the section "Raster Package Bitmap Structure" on page 4-22.

The raster imaging system structure is described on page 4-23.

The raster package control resource type is described on page 6-74 in the chapter "Printing Resources."

PostScript Imaging Messages

QuickDraw GX sends PostScript imaging messages only when the specific driver is printing to a PostScript-based device. The type of device is selected when you specify PostScript imaging in the imaging system resource for your printing extension or printer driver. The imaging system (`'isys'`) resource is described on page 6-33 in the chapter "Printing Resources." The PostScript imaging messages are grouped in this section according to their functionality and you can use them for

- device configuration and control
- device communications
- management of procedure sets
- font management
- document-level structure and formatting control
- page-level structure and formatting control
- imaging control for shapes

GXPostScriptQueryPrinter

QuickDraw GX sends the `GXPostScriptQueryPrinter` message to gather configuration information before imaging begins. You can override the `GXPostScriptQueryPrinter` message to gather information about the current state of the printer and to synchronize communications with it. Your override of the `GXPostScriptQueryPrinter` message must match the following formal declaration:

```
OSErr MyPostScriptQueryPrinter (long *queryResult);
```

`queryResult`
On return, specifies the state of the printer, as shown in Table 4-8.

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

DESCRIPTION

Ater the device connection is opened, QuickDraw GX sends the `GXPostScriptQueryPrinter` message. This message allows QuickDraw GX to gather information from a printer using two-way synchronous communications. The information gathered from the printer is stored in the desktop printer (DTP) configuration file, which all message handlers can access. The state of the printer is returned in the `queryResult` parameter. The values that are returned are shown in Table 4-8.

Table 4-8 Constants for PostScript query results

Constant	Value	Explanation
`gxPrinterOk`	0	The information gathered from the printer is sufficient to initiate imaging of the document. The information is stored in the configuration file.
`gxInitializePrinter`	1	The printer has not been initialized by any version of printing software and therefore must be initialized. After initialization, the information in the configuration file is up to date.
`gxFilePrinting`	2	The output of QuickDraw GX has been redirected to a file, and no immediate two-way synchronous communications is available with the printing device. QuickDraw GX checks that the required information exists in the configuration file and that it is adequate for imaging the document.
`gxResetPrinter`	128	The printer has been initialized with an incompatible version of printing software and therefore must be restarted and initialized. The information in the configuration file is left untouched.

The default implementation of this message verifies whether a communications channel is open to a printer or if the output of the system is being redirected to a file. If the print job is being sent to a printer, the default implementation of `GXPostScriptQueryPrinter` updates the printer configuration information by querying the printer to determine how much memory and which fonts are available on it. The outcome of this message is that the state of the PostScript imaging system is set up to print the job.

You can override the `GXPostScriptQueryPrinter` message to obtain information about the device you need to process the current document. You need to override this message if you want to test the state of an add-on hardware device such as a bin feeder.

SPECIAL CONSIDERATIONS

If you override the `GXPostScriptQueryPrinter` message, you must always forward it to enable the default implementation to return information about the type of connection.

RESULT CODES

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

The default implementation of the `GXPostScriptQueryPrinter` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

GXPostScriptInitializePrinter

QuickDraw GX sends the `GXPostScriptInitializePrinter` message when the printer needs to be initialized. You can override the `GXPostScriptInitializePrinter` message to bring the printer to a known state after it has been restarted. Your override of the `GXPostScriptInitializePrinter` message must match the following formal declaration:

```
OSErr MyPostScriptInitializePrinter (void);
```

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

DESCRIPTION

QuickDraw GX sends the `GXPostScriptInitializePrinter` message when the `GXPostScriptQueryPrinter` message returns an indicator that a device needs to be initialized. The `GXPostScriptQueryPrinter` downloads any code necessary to initialize the printer to a known state.

The default implementation of this message downloads to the printer the latest version (version 7.0) of the LaserWriter PatchPrep procedure set, which defines a collection of PostScript operations that ensure compatibility between the LaserWriter driver and the printer. These operations then become "permanently" available on the printer, meaning that they are available until it is next restarted.

You can override this message if you want to modify the initialization process for the device. For example, you could modify the procedure set to add additional operations that are permanently available on the printer.

SPECIAL CONSIDERATIONS

Although you can totally override the `GXPostScriptInitializePrinter` message, you almost always forward it. Your implementation downloads any commands that you want permanently installed.

RESULT CODES

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

SEE ALSO

The `GXPostScriptQueryPrinter` message is described on page 4-101.

GXPostScriptResetPrinter

QuickDraw GX sends the `GXPostScriptResetPrinter` message when the printer needs to be reset. You can override the `GXPostScriptResetPrinter` message to alter the action taken when a printer is reset. Your override of the `GXPostScriptResetPrinter` message must match the following formal declaration:

```
OSErr MyPostScriptResetPrinter (void);
```

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

DESCRIPTION

QuickDraw GX calls the `GXPostScriptResetPrinter` message when the `GXPostScriptQueryPrinter` message indicates that a device requires resetting. The `GXPostScriptQueryPrinter` message forces a printer to restart itself, resetting it to a rebooted state.

You can override this message to replace the code that is used to reset the printer or to perform other actions when a printer is reset. The default implementation of this message reboots the printer: it first sends the `GXPostScriptExitServer` message and then downloads PostScript code that executes the `quit` operation in the `systemdict` dictionary.

SPECIAL CONSIDERATIONS

You almost always forward the `GXPostScriptResetPrinter` message and then add your own reset actions. If you do totally override this message, you must make sure that your override reboots the printer state.

RESULT CODES

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

The default implementation of the `GXPostScriptResetPrinter` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `GXPostScriptExitServer` message is described in the next section.

The `GXPostScriptQueryPrinter` message is described on page 4-101.

GXPostScriptExitServer

QuickDraw GX sends the `GXPostScriptExitServer` message when the printer needs to be restarted. You can override the `GXPostScriptExitServer` message to change the code used to exit the server loop in the printer. Your override of the `GXPostScriptExitServer` message must match the following formal declaration:

```
OSErr MyPostScriptExitServer (void);
```

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

DESCRIPTION

QuickDraw GX's default implementation of the `GXPostScriptQueryPrinter` message sends the `GXPostScriptExitServer` message when a device requires restarting. This message allows you to modify the permanent state of a printer.

When you exit the server loop, you can make changes that persist after the loop is reentered, such as changing the password. You must send an end-of-file (EOF) to the printer to reenter the server loop.

The default implementation of this message sends the following PostScript code:

```
serverdict begin 000000 exitserver
```

The `000000` value in this code is the default password for the printer.

SPECIAL CONSIDERATIONS

You can either partially or totally override the `GXPostScriptExitServer` message. If you override this message because the password has been changed, you must totally override it.

RESULT CODES

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

The default implementation of the `GXPostScriptExitServer` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `GXPostScriptQueryPrinter` message is described on page 4-101.

GXPostScriptGetStatusText

QuickDraw GX sends the `GXPostScriptGetStatusText` message to query the printer's status channel. You can override the `GXPostScriptGetStatusText` message to add your own handling to the receiving of status strings from the printer. Your override of the `GXPostScriptGetStatusText` message must match the following formal declaration:

```
OSErr MyPostScriptGetStatusText (Handle statusTextHdl);
```

`statusTextHdl`
A handle to the data that is output from the printer's status channel. The first long word in the data is treated as the length of the text.

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

DESCRIPTION

QuickDraw GX sends the `GXPostScriptGetStatusText` message to retrieve the status string from the printer's status channel. This message is sent by the default implementation of the `GXOpenConnection` and `GXDumpBuffer` messages. It informs QuickDraw GX of the current state of a printer.

The default implementation of this message issues a `PAPStatus` call and uses the information stored in the desktop printer to find the device. If the connection is already opened, the default implementation uses the current AppleTalk connection.

QuickDraw GX allocates a handle to 512 bytes for the status information. You can resize this handle as needed to change the size. The format of the handle is as follows:

```
{
   long byteCount;
   char data[];
} **handle;
```

The value of the `byteCount` field must be less than or equal to the size of the handle.

SPECIAL CONSIDERATIONS

You can partially override the `GXPostScriptGetStatusText` message to add special handling to the default implementation. If you are not using a PAP connection, you must totally override this message.

RESULT CODES

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

The default implementation of the `GXPostScriptGetStatusText` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `GXOpenConnection` message is described on page 4-131.

The `GXDumpBuffer` message is described on page 4-142.

GXPostScriptGetPrinterText

QuickDraw GX sends the `GXPostScriptGetPrinterText` message to query the printer's data channel and maintain asynchronous two-way communications with a printer. Your override of the `GXPostScriptGetPrinterText` message must match the following formal declaration:

```
OSErr MyPostScriptGetPrinterText (Handle printerTextHdl);
```

`printerTextHdl`	A handle to the data that is output from the printer's standard output channel. The first long word in the data is treated as the length of the text.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXPostScriptGetPrinterText` message to retrieve any data that the printer sends back on the data channel. This message is sent by the default implementation of the `GXOpenConnection`, `GXFreeBuffer`, and `GXDumpBuffer` messages. It informs QuickDraw GX of the current state of a printer. This message is not sent when printed output is directed to a file.

The default implementation of `GXPostScriptGetPrinterText` message issues a `PAPRead` call on the currently opened AppleTalk connection. If any text is received, it is copied into the handle. If there is no text, the length word in the handle is set to 0.

QuickDraw GX allocates a handle to 512 bytes for the status information. You can resize this handle as needed to change the size. The format of the handle is as follows:

```
{
   long byteCount;
   char data[];
} **handle;
```

The value of the `byteCount` field must be less than or equal to the size of the handle.

SPECIAL CONSIDERATIONS

You can partially override the `GXPostScriptGetPrinterText` message to add special handling to the default implementation. If you are not using a PAP connection, you must totally override this message.

RESULT CODES

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

The default implementation of the `GXPostScriptGetPrinterText` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `GXOpenConnection` message is described on page 4-131.

The `GXDumpBuffer` message is described on page 4-142.

The `GXFreeBuffer` message is described on page 4-143.

An example of using the `GXPostScriptGetPrinterText` and `GXPostScriptScanPrinterText` messages is shown in Listing 4-1.

Listing 4-1 Using the `GXPostScriptGetPrinterText` and `GXPostScriptScanPrinterText` messages

```
Handle response;
   /* download the request and flush the buffers */
   anErr = DownloadResource(kPostScriptType, kPaperTrayQueryID);
   nrequire(anErr, DownloadResource);

   /* flush all buffers to the device */
   anErr = Send_GXWriteData(nil, 0);
   nrequire(anErr, FlushBuffers);

   /* wait for a response from the printer */
   response = NewHandleClear(sizeof(long));

   anErr = MemError();
   nrequire(anErr, GetResponseBuffer);

   for (;;)
      {
      if (anErr = Send_GXPostScriptGetPrinterText(response))
         break;
      if ((** long **) response) > 0
         {
         if (Munger(response, 4, "*", 1, "", 0) > 0)
            break;
         else
            {
            if (anErr=Send_GXPostScriptScanPrinterText(response))
               break;
            }
         }
      }
```

GXPostScriptScanStatusText

QuickDraw GX sends the `GXPostScriptScanStatusText` message after getting a printer status message back from the printer to interpret and prepare the status string. You can override the `GXPostScriptScanStatusText` message if the status string has information you need or if the format of this string is not standard and cannot be interpreted by QuickDraw GX's default implementation. Your override of the `GXPostScriptScanStatusText` message must match the following formal declaration:

```
OSErr MyPostScriptScanStatusText (Handle statusTextHdl);
```

`statusTextHdl`
A handle to the text returned in the `GXPostScriptGetStatusText` message.

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

DESCRIPTION

QuickDraw GX sends the `GXPostScriptScanStatusText` message every time any text is returned from the `GXPostScriptGetStatusText` message (that is, when the message does not return an empty buffer). The `GXPostScriptScanStatusText` message performs error detection and reporting, using the Postscript scan (`'scan'`) resource to interpret the string. Normally, the only text returned by a printer is an error. This message searches the text buffers to determine the printing status and interprets the status string to determine the state of the printing process.

The default implementation of the `GXPostScriptScanStatusText` message searches the text string for special keywords that determine the status of the printing device. It also reformats the string for reading by a user and calls the `GXReportStatus` function to display the string in the desktop printer window.

You need to override this message if the status string holds information that is meaningful only to you or if the device to which the workstation is connected has a format different from that of Apple LaserWriters. Only the Apple LaserWriter format is understood by the default implementation of this message.

SPECIAL CONSIDERATIONS

You can partially or totally override the `GXPostScriptScanStatusText` message.

RESULT CODES

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

The default implementation of the `GXPostScriptScanStatusText` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `GXPostScriptGetStatusText` message is described on page 4-106.

The `GXReportStatus` function is described on page 5-17 in the chapter "Printing Functions for Message Overrides."

GXPostScriptScanPrinterText

QuickDraw GX sends the `GXPostScriptScanPrinterText` message after getting a printer data message back from the printer to interpret and prepare the printer string. You can override the `GXPostScriptScanPrinterText` message if the printer string contains information you wish to use or if the device to which the workstation is connected has a format different from that of Apple LaserWriters. Your override of the `GXPostScriptScanPrinterText` message must match the following formal declaration:

```
OSErr MyPostScriptScanPrinterText (Handle printerTextHdl);
```

`printerTextHdl`
A handle to the text returned in the `GXPostScriptGetPrinterText` message.

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

DESCRIPTION

QuickDraw GX sends the `GXPostScriptScanPrinterText` message every time any text is returned from the `GXPostScriptGetPrinterText` message (that is, when this message does not return an empty buffer). The `GXPostScriptScanPrinterText` message interprets the text of the printer string to determine the state of the printing process. It performs error detection and reporting, using the Postscript scan (`'scan'`) resource to interpret the string.

The default implementation of this message searches the text string for special keywords that determine the status of the printing device. It also reformats the string for reading by a user and calls the `GXReportStatus` function to display the string in the desktop printer window.

You need to override this message when part or all of the printer string holds some piece of information that is only meaningful to you or if the device to which the workstation is connected has a format different from that of Apple LaserWriters. Only the Apple LaserWriter format is understood by the default implementation of this message.

SPECIAL CONSIDERATIONS

You can partially or totally override the `GXPostScriptScanPrinterText` message.

RESULT CODES

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

The default implementation of the `GXPostScriptScanPrinterText` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `GXPostScriptGetPrinterText` message is described on page 4-107.

An example of using the `GXPostScriptGetPrinterText` and `GXPostScriptScanPrinterText` messages is shown in Listing 4-1 on page 4-109.

The `GXReportStatus` functon is described on page 5-17 in the chapter "Printing Functions for Message Overrides."

GXPostScriptGetDocumentProcSetList

QuickDraw GX sends the `GXPostScriptGetDocumentProcSetList` message at the start of imaging for a docuemtn to determine which procedure sets are needed for the document. You can override the `GXPostScriptGetDocumentProcSetList` message to retrieve information about those procedure sets. Your override of the `GXPostScriptGetDocumentProcSetList` message must match the following formal declaration:

```
OSErr MyPostScriptGetDocumentProcSetList (
                        gxProcSetListHdl procSetListHdl,
                        gxPostScriptImageDataHandle hImageData);
```

`procSetListHdl`	On entry, a handle to a preallocated PostScript procedure set list structure. On return, a value that specifies the procedure sets needed to image the document.
`hImageData`	A handle to the PostScript imaging system structure.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXPostScriptGetDocumentProcSetList` message when you send the `GXImageDocument` message. The `GXPostScriptGetDocumentProcSetList` message gathers information on which procedure sets are needed to image a specified document.

The default implementation of this message returns the procedure sets needed by the PostScript generic driver, the PostScript imaging engine, and the font handler. The set of procedure sets for all documents are typically the same: those that are specified in the imaging system's structure, as determined when the printer is initially queried. However, some documents might require special handling, which means additional procedure sets need to be downloaded.

You can then use the `GXPostScriptDownloadProcSetList` message to make these procedures available in the PostScript device.

SPECIAL CONSIDERATIONS

You must forward the `GXPostScriptGetDocumentProcSetList` message.

RESULT CODES

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

SEE ALSO

The PostScript procedure set list structure is described on page 4-29.

The PostScript imaging system structure is described on page 4-26.

The `GXImageDocument` message is described on page 4-93.

The `GXPostScriptDownloadProcSetList` message is described in the next section.

An example of overriding the `GXPostScriptGetDocumentProcSetList` message is shown in Listing 4-2. This example downloads a procedure set that corrects a known bug in the PostScript implementation in the LaserWriter IIg and IIf printers.

Listing 4-2 An example of the `GXPostScriptGetDocumentProcSetList` message

```
OSErr DriverGetDocumentProcSetList (gxProcSetListHdl procSetList,
                            gxPostScriptImageDataHdl imageDataHdl)
{
   OSErr anErr;

   anErr = Forward_GXPostScriptGetDocumentProcSetList(
                                    procSetList, imageDataHdl);
   if (anErr == noErr)
      {
      /*
         Send this proc set to the IIg or IIf, or for portable
         PostScript.
      */
      if
         (
         ((**imageDataHdl).renderOptions &
                                  gxPortablePostScriptOption) ||
         (gPrinterType == kLWIIf) ||
         (gPrinterType == kLWIIg)
         )
         {
         SetHandleSize( (Handle) procSetList,GetHandleSize(
                (Handle) procSetList) + sizeOf(ProcSetListRec));
         anErr == MemError();
         if (anErr == noErr)
            {
            gxProcSetListPtr pList = (gxProcSetListPtr)
               ((unsigned long) (*procSetList)
                + GetHandleSize(Handle) procSetList)
                - sizeOf(gxProcSetListRec);
            pList->clientid = 'drvr';
            pList->controlType = gxPostscriptProcSetControlType;
            pList->controlid = kFAndGShowPatch;
            pList->dataType = 'rdws';
            }
         }
      }
   return(anErr);
}
```

GXPostScriptDownloadProcSetList

QuickDraw GX sends the `GXPostScriptDownloadProcSetList` message to guarantee that the needed procedure sets are available on the printer. You can override the `GXPostScriptDownloadProcSetList` message to modify the list of procedure sets that are needed to print a document. Your override of the `GXPostScriptDownloadProcSetList` message must match the following formal declaration:

```
OSErr MyPostScriptDownloadProcSetList (
                        gxProcSetListHdl procSetListHdl,
                        gxPostScriptImageDataHandle hImageData);
```

`procSetListHdl`
: A handle to a PostScript procedure set list structure. This handle is filled by the `GXPostScriptGetDocumentProcSetList` message.

`hImageData`
: A handle to the PostScript imaging system structure.

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

DESCRIPTION

QuickDraw GX sends the `GXPostScriptDownloadProcSetList` message to guarantee that each of the procedure sets in the procedure set list is available in the printer. QuickDraw GX first sends the `GXPostScriptGetDocumentProcSetList` message to determine which procedure sets are needed for the document, and then sends this message to make sure that the procedure sets are downloaded to the printer.

The default implementation of the `GXPostScriptDownloadProcSetList` message downloads each of the procedure sets to the printer. You can override this message if you want to change the list of procedure sets that are going to be downloaded to the printer.

SPECIAL CONSIDERATIONS

You must forward the `GXPostScriptDownloadProcSetList` message.

RESULT CODES

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

The default implementation of the `GXPostScriptDownloadProcSetList` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The PostScript procedure set list structure is described on page 4-29.

The PostScript imaging system structure is described on page 4-26.

The `GXPostScriptGetDocumentProcSetList` message is described in the previous section.

GXPostScriptGetPrinterGlyphsInformation

QuickDraw GX sends the `GXPostScriptGetPrinterGlyphsInformation` message to allow a printing extension or printer driver to communicate with the imaging system about the fonts and glyphs that are resident in the output device. You can override the `GXPostScriptGetPrinterGlyphsInformation` message if you are doing your own font management. Your override of the `GXPostScriptGetPrinterGlyphsInformation` message must match the following formal declaration:

```
OSErr MyPostScriptGetPrinterGlyphsInformation (
                                gxPrinterGlyphsRec *glyphPtr);
```

`glyphPtr`	A pointer to a PostScript glyphs structure.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The PostScript imaging system sends the `GXPostScriptGetPrinterGlyphsInformation` message before it starts to image a document. It uses the information gathered by this message to determine which fonts or which glyphs in a font need to be downloaded to the printer for the document to be printed.

On entry to this message, the `theFont` field of the PostScript glyphs structure, of data type `gxPrinterGlyphsRec`, must be filled in with a valid font reference to the font for which information is desired. You can fill in the `platform`, `script`, and `language` fields of the PostScript glyphs structure to tell the imaging system which glyphs from a font are present in the printer.

If the `platform` field of the structure has the value `-1`, then the `script` and `language` field values are ignored, and the `glyphBits` array is filled in with the glyphs that are actually present for the font.

If the `platform` field of the structure has any value other than `-1`, then the `platform`, `script`, and `language` fields together define how the imaging system maps glyphs into the printer's encoding scheme.

The default implementation of the `GXPostScriptGetPrinterGlyphsInformation` message uses the information that was gathered by the `GXPostScriptQueryPrinter` message to fill out the PostScript glyphs structure. It also looks for a `'pfnt'` resource that has the same font name as that in the `theFont` field of the structure and gathers information from that resource.

SPECIAL CONSIDERATIONS

You must forward the `GXPostScriptGetPrinterGlyphsInformation` message.

RESULT CODES

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

The default implementation of the `GXPostScriptGetPrinterGlyphsInformation` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The PostScript glyphs structure is described on page 4-28.

The `GXPostScriptQueryPrinter` messages is described on page 4-101.

The PostScript printer font (`'pfnt'`) resource is described on page 6-84 in the chapter "Printing Resources."

For more information about how the imaging system maps glyphs, read about the `'cmap'` table in the book *Inside Macintosh: QuickDraw GX Typography.*

GXPostScriptStreamFont

QuickDraw GX sends the `GXPostScriptStreamFont` message when the imaging system determines that a font is needed to print a document. You can override the `GXPostScriptStreamFont` message to provide font management on a remote host. Your override of the `PostScriptStreamFont` message must match the following formal declaration:

```
OSErr MyPostScriptStreamFont (gxFont fontRef,
                              gxScalerStream *stream);
```

`fontRef`	A reference to the font to be streamed.
`stream`	A pointer to the `gxScalerStream` structure that specifies how to stream the font to the printer.

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

DESCRIPTION

The default implementation of the `GXPostScriptStreamFont` message streams the specified font to the printer. You can override this message to determine when a font is being downloaded or to change the streaming method used to send the font to the printer.

SPECIAL CONSIDERATIONS

You must forward the `GXPostScriptStreamFont` message.

RESULT CODES

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

The default implementation of the `GXPostScriptStreamFont` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `gxScalerStream` structure is described in *Inside Macintosh: QuickDraw GX Environment and Utilities.*

GXPostScriptDoDocumentHeader

QuickDraw GX sends the `GXPostScriptDoDocumentHeader` message to generate the PostScript program header for a document. You can override the `GXPostScriptDoDocumentHeader` message to change the PostScript document structure and formatting setup for a document. Your override of the `GXPostScriptDoDocumentHeader` message must match the following formal declaration:

```
OSErr MyPostScriptDoDocumentHeader (
                                gxPostScriptImageDataHandle hImageData);
```

`hImageData`
A handle to the PostScript imaging system structure.

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

DESCRIPTION

QuickDraw GX sends the `GXPostScriptDoDocumentHeader` message during its default implementation of the `GXImageDocument` message when printing to a PostScript device. The `GXPostScriptDoDocumentHeader` message is the first message sent when generating a PostScript program. QuickDraw GX uses this message to generate a PostScript program header, which is required for conforming PostScript programs. The information for the header is obtained from the job object.

You can override this message to build your own document header or to add comments to a PostScript header.

SPECIAL CONSIDERATIONS

You can totally override the `GXPostScriptDoDocumentHeader` message to build your own PostScript program header, or you can forward this message and then add your own information to the header created by the default implementation.

RESULT CODES

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

The default implementation of the `GXPostScriptDoDocumentHeader` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `GXImageDocument` message is described on page 4-93.

The PostScript imaging system structure is described on page 4-26.

GXPostScriptDoDocumentSetup

QuickDraw GX sends the `GXPostScriptDoDocumentSetup` message to set up document-level formatting information. You can override the `GXPostScriptDoDocumentSetup` message to issue any code that is required for the imaging of the whole document. Your override of the `GXPostScriptDoDocumentSetup` message must match the following formal declaration:

```
OSErr MyPostScriptDoDocumentSetup (
                        gxPostScriptImageDataHandle hImageData);
```

`hImageData`
A handle to the PostScript imaging system structure.

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

DESCRIPTION

QuickDraw GX sends the `GXPostScriptDoDocumentSetup` message in its default implementation of the `GXImageDocument` message, prior to imaging the first page of the document. You can override this message to issue any PostScript instructions that apply to the document; for example, to set up the default halftone accuracy.

The default implementation of this message changes the display in the desktop printer window to reflect the current document and user names.

SPECIAL CONSIDERATIONS

You always forward the `GXPostScriptDoDocumentSetup` message to allow other message handlers the opportunity to perform document-setup tasks. Forward this message and then add your own operations.

RESULT CODES

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

The default implementation of the `GXPostScriptDoDocumentSetup` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The PostScript imaging system structure is described on page 4-26.

The `GXImageDocument` message is described on page 4-93.

GXPostScriptDoDocumentTrailer

QuickDraw GX sends the `GXPostScriptDoDocumentTrailer` message to generate the document trailer that is appended to the end of the PostScript program for a document. You can override the `GXPostScriptDoDocumentTrailer` message to perform an action after all pages in a document have been imaged. Your override of the `GXPostScriptDoDocumentTrailer` message must match the following formal declaration:

```
OSErr MyPostScriptDoDocumentTrailer (
                                gxPostScriptImageDataHandle hImageData);
```

`hImageData`	A handle to the PostScript imaging system structure.

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

DESCRIPTION

QuickDraw GX sends the `GXPostScriptDoDocumentTrailer` message to generate the document trailer after terminating the imaging of a document. It sends this message during its default implementation of the `GXImageDocument` message, after all the pages of the document have been imaged.

The default implementation of this message changes the display in the desktop printer window to reflect the number of pages printed for the document.

SPECIAL CONSIDERATIONS

You always forward the `GXPostScriptDoDocumentTrailer` message to allow other message handlers the opportunity to perform document-termination tasks.

RESULT CODES

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

The default implementation of the `GXPostScriptDoDocumentTrailer` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `GXImageDocument` message is described on page 4-93.

The PostScript imaging system structure is described on page 4-26.

GXPostScriptDoPageSetup

QuickDraw GX sends the `GXPostScriptDoPageSetup` message to set up page-level formatting information. You can override the `GXPostScriptDoPageSetup` message to customize the page setup information. Your override of the `GXPostScriptDoPageSetup` message must match the following formal declaration:

```
OSErr MyPostScriptDoPageSetup (gxFormat aFormat, long pageIndex,
                        gxPostScriptImageDataHandle hImageData);
```

`aFormat`	The format object for the page to be printed.
`pageIndex`	The number of the page that is about to be imaged.

`hImageData`	A handle to the PostScript imaging system structure.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXPostScriptDoPageSetup` message after a page has been spooled and before it is imaged. This message performs all necessary actions for setting up the page in the printer, including initiating the `GXPostScriptSelectPaperType` message.

The default implementation of this message takes the page-formatting information and translates it into PostScript code. You can override this message if you need to perform any special actions for specific page numbers.

SPECIAL CONSIDERATIONS

You always forward the `GXPostScriptDoPageSetup` message to allow other message handlers the opportunity to perform page setup tasks.

RESULT CODES

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

The default implementation of the `GXPostScriptDoPageSetup` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The PostScript imaging system structure is described on page 4-26.

The `GXPostScriptSelectPaperType` message is described in the next section.

GXPostScriptSelectPaperType

QuickDraw GX sends the `GXPostScriptSelectPaperType` message prior to imaging the page, to select the printer's paper type for the page. You can override the `GXPostScriptSelectPaperType` message to do anything you need to do to make sure the device is set up correctly. Your override of the `GXPostScriptSelectPaperType` message must match the following formal declaration:

```
OSErr MyPostScriptSelectPaperType (gxPaperType aPaperType,
         long pageIndex, gxPostScriptImageDataHandle hImageData);
```

`aPaperType`	The paper-type object that is needed.
`pageIndex`	The page number of the page that is about to be imaged.
`hImageData`	A handle to the PostScript imaging system structure.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The PostScript generic driver invokes the `GXPostScriptSelectPaperType` message when it is setting up the format for a page—after the page has been despooled but before it is imaged. This message selects the paper type for a printer. This message is sent by the default implementation of the `GXPostScriptDoPageSetup` message.

The default implementation of this message looks for a collection item of type `'post'` in the paper-type collection and sends that to the printer as the paper type for the page. The collection item must be valid PostScript that works for all PostScript devices.

You can override this message to customize the way that setting the paper type for the page works.

SPECIAL CONSIDERATIONS

You usually forward the `GXPostScriptSelectPaperType` message to other message handlers; however, if you are overriding this message to issue your own `setpagedevice` operation, do not forward this message.

RESULT CODES

`gxSegmentLoadFailedErr`	A required code segment could not be found, or there was not enough memory to load it.
`gxPrUserAbortErr`	The user has canceled printing.
`gxPaperTypeNotFound`	The specified paper type could not be found.
`gxNoSuchPTGroup`	The specified paper type could not be found.

The default implementation of the `GXPostScriptSelectPaperType` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The PostScript imaging system structure is described on page 4-26.

Paper types and paper-type collection items are described in *Inside Macintosh: QuickDraw GX Printing*.

GXPostScriptDoPageTrailer

QuickDraw GX sends the `GXPostScriptDoPageTrailer` message to generate the page trailer that is appended to the end of the PostScript program for a page. You can override the `GXPostScriptDoPageTrailer` message if you need to add comments to the page trailer. Your override of the `GXPostScriptDoPageTrailer` message must match the following formal declaration:

```
OSErr MyPostScriptDoPageTrailer (
                        gxPostScriptImageDataHandle hImageData);
```

`hImageData`	A handle to the PostScript imaging system structure.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXPostScriptDoPageTrailer` message at the end of every page to provide any features requiring control.

The default implementation of the `GXPostScriptDoPageTrailer` message sends the `GXPostScriptEjectPage` message to eject a page. You can override this message to create a PostScript page trailer.

SPECIAL CONSIDERATIONS

You usually forward the `GXPostScriptDoPageTrailer` message to allow other message handlers the opportunity to perform page-termination tasks and so that the default implementation can perform its operations.

RESULT CODES

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

The default implementation of the `GXPostScriptDoPageTrailer` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The PostScript imaging system structure is described on page 4-26.

The `GXPostScriptEjectPage` message is described in the next section.

GXPostScriptEjectPage

QuickDraw GX sends the `GXPostScriptEjectPage` message when printing of a page is done and the printer is ready for the next page. You can override the `GXPostScriptEjectPage` message to perform any actions required after a page has been printed and before the next page begins. Your override of the `GXPostScriptEjectPage` message must match the following formal declaration:

```
OSErr MyPostScriptEjectPage (gxPaperType aPaperType,
                long pageIndex, long copiesCount, short erasePage,
                gxPostScriptImageDataHandle hImageData);
```

`aPaperType`	The paper-type object for the page that was just printed.
`pageIndex`	The page number of the page that was just printed.
`copiesCount`	The number of copies of the page needed.
`erasePage`	A value that indicates whether the printer's frame buffer needs to be erased when the page is ejected.
`hImageData`	A handle to the PostScript imaging system structure.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXPostScriptEjectPage` message when the imaging system has completed imaging tasks for a single page and is ready for the next page.

You can specify whether the page image is to be erased from the printer's frame buffer in the `erasePage` parameter. If this value is `true`, the imaged is erased from the page buffer. If the difference between the page that was just printed and the next page is minimal, not erasing the frame buffer can significantly improve the printing speed. This is often used when printing forms.

The default implementation of this message uses the PostScript `showpage` operator when the page image needs to be erased and uses the PostScript `copypage` operator when the page image is not to be erased.

SPECIAL CONSIDERATIONS

You usually forward the `GXPostScriptEjectPage` message to allow other message handlers the opportunity to determine if the frame buffer needs to be erased.

RESULT CODES

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

The default implementation of the `GXPostScriptEjectPage` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The PostScript imaging system structure is described on page 4-26.

GXPostScriptProcessShape

QuickDraw GX sends the `GXPostScriptProcessShape` message when converting a shape object into PostScript. You can override the `GXPostScriptProcessShape` message to exercise imaging control for a shape. Your override of the `GXPostScriptProcessShape` message must match the following formal declaration:

```
OSErr MyPostScriptProcessShape (gxShape aShape, long count,
                               gxTransform list[]);
```

`aShape`	The shape object that is to be printed.
`count`	The number of transform objects in the transform list.
`list`	A list of transforms through which the shape needs to be mapped when converted to PostScript.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXPostScriptProcessShape` message to provide you with imaging control for individual shapes.

The default implementation of this message takes a shape and produces the PostScript code needed to image the shape.

You can override this message to change the way that a shape is imaged on a PostScript device.

Note
QuickDraw GX internally tracks the state of the printer. If you override this message, then QuickDraw GX's version of the printer's state can become out of sync with the hardware. This means that if you override this message for any shapes, you need to override it for all shapes. u

SPECIAL CONSIDERATIONS

You can modify the shape or the transform list before forwarding the `GXPostScriptProcessShape` message, or you can totally override the `GXPostScriptProcessShape` message to change the PostScript code that is generated for the shape.

RESULT CODES

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

Vector Imaging Messages

QuickDraw GX sends vector imaging messages only when the specific driver is printing for a vector-based device. The type of device is selected using the imaging system (`'isys'`) resource, which is described on page 6-33 in the chapter “Printing Resources.”

GXVectorPackageShape

QuickDraw GX sends the `GXVectorPackageShape` message to translate a shape into a package that is sent to the printing device. You need to override the `GXVectorPackageShape` message to turn a shape into the appropriate pen commands. Your override of the `GXVectorPackageShape` message must match the following formal declaration:

```
OSErr MyVectorPackageShape (gxShape aShape, long penIndex);
```

`aShape`	The shape object to be packaged.
`penIndex`	The index of the pen to use to draw the shape.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The default implementation of the `GXRenderPage` message for the vector imaging system sends the `GXVectorPackageShape` message to translate a QuickDraw GX shape into a package and send it to a printing device.

The default implementation of this message does nothing. Your override needs to convert the shape into a series of pen commands and send those commands to your printing device with the `GXBufferData` or `GXWriteData` messages.

QuickDraw GX converts high-level shapes such as bitmaps, paths, curves, glyphs, text, and text layout into lower-level vector-type shapes such as polygons, rectangles, and lines. For all shapes, QuickDraw GX color sorts, applies transfer modes if needed, clips them against overlapping shapes, and resolves styles, inks, and transforms. It converts the resultant shape into a lower-level shape, if necessary, and then sends the `GXVectorPackageShape` message.

Your override of the `GXVectorPackageShape` message needs to convert the shape sent by QuickDraw GX into a device-specific language such as HPGL. You can also override this message to perform other tasks, such as supporting a preview feature in which data is displayed on the user's screen and written to a file before, or instead of, being sent to the device.

SPECIAL CONSIDERATIONS

You never send the `GXVectorPackageShape` message yourself.

You always totally override the `GXVectorPackageShape` message.

RESULT CODES

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

SEE ALSO

The `GXRenderPage` message is described on page 4-96.

The `GXBufferData` message is described on page 4-139.

The `GXWriteData` message is described on page 4-141.

GXVectorLoadPens

QuickDraw GX sends the `GXVectorLoadPens` message when drawing with the pens currently in the carousel is finished and it is time to move on to the next carousel. You can override the `GXVectorLoadPens` message to inform QuickDraw GX of the number and position of pens in the plotter. Your override of the `GXVectorLoadPens` message must match the following formal declaration:

```
OSErr MyVectorLoadPens (gxPenTableHdl penTable,
                        long *shapeCounts, Boolean *penTableChanged);
```

`penTable`	A handle to the pen table. This structure is described in the section "Vector Pen Table Structure" beginning on page 4-34.

`shapeCounts`
: A pointer to an array of shape counts that corresponds to the pens in the pen table.

`*penTableChanged`
: A pointer to a Boolean value. On return, the value is `true` if the pen table is modified and `false` if it remains the same.

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

DESCRIPTION

QuickDraw GX's default implementation of the `GXRenderPage` message for the vector imaging system sends the `GXVectorLoadPens` message when all imaging has taken place for the pens in the current pen carousel and it is time to move on to the next carousel. You can use the `GXVectorLoadPens` message to prompt the user to change pens loaded in the carousel.

The default implementation of this message does nothing. Your override of `GXVectorLoadPens` needs to mark all of the currently loaded pens as not loaded and prompt the user to install the new pen carousel. You then update the positions of the pens in the pen table to reflect their positions in the plotter. If you do not override this message, the number of pens in the pen table must be the same as what is currently loaded in the carousel.

Note

The `GXVectorLoadPens` message should only update the pen position field of the pens. It should not rearrange pens or remove, add, or change any other fields of the pen table. u

SPECIAL CONSIDERATIONS

You never send the `GXVectorLoadPens` message yourself.

You always totally override the `GXVectorLoadPens` message.

RESULT CODES

`gxSegmentLoadFailedErr`	A required code segment could not be found, or there was not enough memory to load it.
`gxPrUserAbortErr`	The user has canceled printing.
`gxInvalidPenTable`	The specified pen table is not valid.

SEE ALSO

The `GXRenderPage` message is described on page 4-96.

The vector pen table structure is described on page 4-34.

GXVectorVectorizeShape

QuickDraw GX sends the `GXVectorVectorizeShape` message to convert a complex shape into simpler, device-level shapes such as polygons. You can override the `GXVectorVectorizeShape` message to change the plotting of particular shapes. Your override of the `GXVectorVectorizeShape` message must match the following formal declaration:

```
OSErr MyVectorVectorizeShape (gxShape aShape, long penIndex,
                  gxVectorShapeDataRec *aVectorShapeDataRec);
```

`aShape`	The shape object to be vectorized.
`penIndex`	The index of the pen to draw the shape.
`aVectorShapeDataRec`	On return, a pointer to a vector shape structure that contains the vector data created from the shape.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX's default implementation of `GXRenderPage` sends the `GXVectorVectorizeShape` message when a shape is ready to be converted into simpler device-level shapes. It sends the message to vectorize a specified shape (convert it into line segments). It sends the message after all transfer modes are resolved and all clipping and transforms have been concatenated.

QuickDraw GX's default implementation of this message converts shapes into polygons.

You can override this message to effect the plotting of particular shape types. You may find this useful for plotters that can handle certain shapes more easily than the shapes provided by the default implementation. You can completely remove undesired shapes, handle certain shapes entirely, modify shapes prior to forwarding the message, and send some shapes through unmodified.

The `GXVectorVectorizeShape` message is initiated before any style mappings are applied to the shape and before it is converted into a low-level shape. The message is invoked after clipping, color sorting, and transfer-mode resolution are complete. Before the message is sent, all the parent clips and mappings are concatenated with the shape's clip and mapping. The shape is left as close to its original form as possible. For example, text remains text shape and is not converted into a path.

You can override this message to customize rendering of certain shapes on your vector device. For example, if you prefer to use the device's native fonts to plot text, glyph, and layout shapes, you can override this message. Note that it is your responsibility to apply style and transforms to the shape.

SPECIAL CONSIDERATIONS

You never send the `GXVectorVectorizeShape` message yourself.

You can totally override the `GXVectorVectorizeShape` message, or you can partially override it to modify the default processing of certain shape types.

RESULT CODES

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

SEE ALSO

The `GXRenderPage` message is described on page 4-96.

The vector shape structure is described on page 4-31.

Device Communications Messages

QuickDraw GX sends device communications messages to send data to a device during the device communications phase of printing. It is very important to note that only in your overrides of these messages can you actually communicate with the device or report problems about the device to the user.

GXOpenConnection

QuickDraw GX sends the `GXOpenConnection` message to establish communications with a device in preparation for sending data to it. You can override the `GXOpenConnection` message to open a connection with a specific device. Your override of the `GXOpenConnection` message must match the following formal declaration:

```
OSErr GXOpenConnection (void);
```

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

DESCRIPTION

QuickDraw GX's default implementation of the `GXImageJob` message sends the `GXOpenConnection` message to prepare to send data to the device.

The default implementation of the `GXOpenConnection` message handles PAP, Serial, and not-connected connections. It reads the resource from the communications (`'comm'`) resource in the desktop printer file.

You can override this message to open a connection with a specific device (such as a sheetfeeder) or to perform status checks at the time the device is opened. If you perform a status check and a failure occurs at that time, you must call the `GXCleanupOpenConnection` function.

Note
The default implementation of this message does not support SCSI devices. u

SPECIAL CONSIDERATIONS

You never send the `GXOpenConnection` message yourself.

If you are implementing a nonsupported type of communications connection, you need to perform a total override of the `GXOpenConnection` message. Otherwise, you need to first forward the message and then perform status checking on the device.

RESULT CODES

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

The default implementation of the `GXOpenConnection` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `GXImageJob` message is described on page 4-90.

You can find an example of an override of the `GXOpenConnection` message in Listing 3-8 on page 3-30 in the chapter “Printer Drivers.”

The `GXCleanupOpenConnection` function is described on page 5-36 in the chapter “Printing Functions for Message Overrides.”

The communications (`'comm'`) resource is described in the section “The Communications ('comm') Resource” beginning on page 6-36 in the chapter “Printing Resources.”

GXOpenConnectionRetry

QuickDraw GX sends the `GXOpenConnectionRetry` message if opening the connection fails. You can override the `GXOpenConnectionRetry` message to try again to open the connection for a device. Your override of the `GXOpenConnectionRetry` message must match the following formal declaration:

```
OSErr MyOpenConnectionRetry (ResType commType, void *commData,
                             Boolean *pRetry, OSErr saveErr);
```

`commType`	The resource type of the communications (`'comm'`) resource to use. The possible values are shown in Table 4-9.
`commData`	A pointer to the communications resource that is stored with the desktop printer.
`pRetry`	On return, the Boolean value is `true` if QuickDraw GX needs to send the `GXOpenConnection` message again and `false` if not.
`saveErr`	The error code generated when the `GXOpenConnection` message failed.

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

DESCRIPTION

QuickDraw GX sends the `GXOpenConnectionRetry` message when a message handler calls the `GXOpenConnection` and it returns an error. You can override this message when you are implementing a communications protocol for a printer that can be shared by multiple users and accepts only one connection at a time. Your override can interpret the error that is specified in the `saveErr` parameter as an indication that the printer is in use by another user and that the connection attempt needs to be tried again.

The type of the communications resource, which you specify in the `commType` parameter, is one of the values shown in Table 4-9.

Table 4-9 Communications resource types

Constant	Value	Explanation
`Serial`	`'SPTL'`	Serial communications resource
`PAP`	`'PPTL'`	AppleTalk PAP communications resource
`SCSI`	`'sPTL'`	SCSI communications resource
`PrinterShare`	`'ptsr'`	PrinterShare communications resource
`NotConnected`	`'Nops'`	No communications resource

The default implementation of the `OpenRetryConnection` message sets the value of the `pRetry` return parameter to `false` for most devices. The exception is for PAP devices that are using the PostScript imaging system. In this case, the default implementation sets the `pRetry` parameter value to `true` if it determines that the printer is busy with another user’s print job.

If your device automatically handles retrying connections, override this message and set the `pRetry` parameter to `true`.

SPECIAL CONSIDERATIONS

You never send the `GXOpenConnectionRetry` message yourself.

If you are implementing a nonsupported type of communications connection, you need to perform a total override of the `GXOpenConnectionRetry` message. Otherwise, you need to first forward the message and then perform your tasks.

RESULT CODES

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

SEE ALSO

The `GXOpenConnection` message is described in the previous section.

The communications (`'comm'`) resource is described in the section "The Communications ('comm') Resource" beginning on page 6-36 in the chapter "Printing Resources."

GXCleanupOpenConnection

QuickDraw GX sends the `GXCleanupOpenConnection` message when an operation that has to be undone fails during the processing of an `GXOpenConnection` message. You need to override the `GXCleanupOpenConnection` message if you perform operations that must be undone after a failure in `GXOpenConnection`. Your override of the `GXCleanupOpenConnection` message must match the following formal declaration:

```
void MyCleanupOpenConnection (void);
```

DESCRIPTION

When an operation fails in your override of the `GXOpenConnection` message, you need to call the `GXCleanupOpenConnection` function.

When you call the `GXCleanupOpenConnection` function, QuickDraw GX sends the `GXCleanupOpenConnection` message to any message handlers that follow your printing extension or printer driver in the message chain. Each message handler is responsible for cleaning up any operations that it performed in its `GXOpenConnection` override. Usually this involves deallocating any storage that you allocated in your override of the `GXOpenConnection` message to open the connection.

The `GXCleanupOpenConnection` message follows the same path through the message chain as did the original `GXOpenConnection` message, which allows the cleaning up to occur in the correct order.

The default implementation of the `GXCleanupOpenConnection` message disposes of memory allocated by the default implementation of the `GXOpenConnection` message.

SPECIAL CONSIDERATIONS

You never send the `GXCleanupOpenConnection` message yourself; however, you can call the `GXCleanupOpenConnection` function, which then sends this message.

You need to forward the `GXCleanupOpenConnection` message so that other message handlers can perform their cleanup tasks.

SEE ALSO

The `GXCleanupOpenConnection` function is described on page 5-36 in the chapter "Printing Functions for Message Overrides" in this book.

The `GXOpenConnection` message is described in the previous section.

GXCloseConnection

QuickDraw GX sends the `GXCloseConnection` message when all of the data for a print job has been sent to the device. You can override the `GXCloseConnection` message to perform actions your printing extension or printer driver requires at the completion of a job. Your override of the `GXCloseConnection` message must match the following formal declaration:

```
OSErr MyCloseConnection (void);
```

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

DESCRIPTION

The default implementation of the `GXImageJob` message sends the `GXCloseConnection` message at the end of sending to a device all of the data of a print job.

The default implementation of the `GXCloseConnection` message closes the connection to a printer. It handles PAP, serial, and not-connected connections. You can override this message to close a connection with a specific device, such as a sheetfeeder.

SPECIAL CONSIDERATIONS

You never send the `GXCloseConnection` message yourself.

If you are implementing a nonsupported type of communications connection, you need to perform a total override of the `GXCloseConnection` message. Otherwise, you need to perform your tasks and then forward the message.

RESULT CODES

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

The default implementation of the `GXCloseConnection` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `GXImageJob` message is described on page 4-90.

GXStartSendPage

QuickDraw GX sends the `GXStartSendPage` message when it's ready to send a page to the printer. You can override the `GXStartSendPage` message to set up printing for the next page. Your override of the `GXStartSendPage` message must match the following formal declaration:

```
OSErr MyStartSendPage (gxFormat pageFormat);
```

`pageFormat`	The format for this page.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX's default implementation of the `GXImagePage` message sends the `GXStartSendPage` message to signal that QuickDraw GX is ready to start sending a page to the printer.

The default implementation of this message handles PAP, serial, and not-connected connections. You can override this message to perform tasks such as making sure that the previous page printed successfully and resetting page margins in the printer.

You must forward the `GXStartSendPage` message to other message handlers so that they can override it. If your override fails, you need to call the `GXCleanupStartSendPage` function to notify other handlers of the failure. If another handler returns an error, you must undo anything that you've done and return the same error.

SPECIAL CONSIDERATIONS

You never send the `GXStartSendPage` message yourself.

If you are implementing a nonsupported type of communications connection, you need to perform a total override of the `GXStartSendPage` message. Otherwise, you need to first forward the message and then perform your tasks.

RESULT CODES

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

The default implementation of the `GXStartSendPage` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

You can find an example of an override of the `GXStartSendPage` message in Listing 3-13 on page 3-39 in the chapter "Printer Drivers."

The `GXCleanupStartSendPage` function is described on page 5-37 in the chapter "Printing Functions for Message Overrides."

The `GXImagePage` message is described on page 4-94.

GXCleanupStartSendPage

QuickDraw GX sends the `GXCleanupStartSendPage` message when an operation that has to be undone fails during the processing of a `GXStartSendPage` message. You need to override the `GXCleanupStartSendPage` message if you have overridden the `GXStartSendPage` message and your override contains code that can fail, such as a memory allocation or device initialization. Your override of the `GXCleanupStartSendPage` message must match the following formal declaration:

```
void MyCleanupStartSendPage (void);
```

DESCRIPTION

When an operation fails in your override of the `GXStartSendPage` message, you need to call the `GXCleanupStartSendPage` function.

When you call the `GXCleanupStartSendPage` function, QuickDraw GX sends the `GXCleanupStartSendPage` message to any message handlers that follow your printing extension or printer driver in the message chain. Each message handler is responsible for cleaning up any operations that it performed in its `GXStartSendPage` override. Usually this involves deallocating any storage that you allocated in your override of the `GXStartSendPage` message.

The `GXCleanupStartSendPage` message follows the same path through the message chain as did the original `GXStartSendPage` message, which allows the cleaning up to occur in the correct order.

The default implementation of the `GXCleanupStartSendPage` message disposes of memory allocated by the default implementation of the `GXStartSendPage` message.

SPECIAL CONSIDERATIONS

You never send the `GXCleanupStartSendPage` message yourself; however, you can call the `GXCleanupStartSendPage` function, which then sends this message.

You must forward the `GXCleanupStartSendPage` message to allow other message handlers to clean up any storage that they allocated in response to the `GXStartSendPage` message.

SEE ALSO

The `GXCleanupStartSendPage` function is described on page 5-37 in the chapter "Printing Functions for Message Overrides."

The `GXStartSendPage` message is described in the previous section.

GXFinishSendPage

QuickDraw GX sends the `GXFinishSendPage` message after all of the data for a page has been sent to the printer. You can override the `GXFinishSendPage` message to perform some action at the end of a page. Your override of the `GXFinishSendPage` message must match the following formal declaration:

```
OSErr MyFinishSendPage (void);
```

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

DESCRIPTION

QuickDraw GX's default implementation of the `GXImagePage` message sends the `GXFinishSendPage` message after the page is sent to the printer and before it starts sending the next page.

The default implementation of the `GXFinishSendPage` message handles PAP, serial, and not-connected connections. You override this message to perform some action at the end of the page, such as telling the printer to print the page that is loaded into its memory or prompting the user to add paper for a manual-feed print job.

SPECIAL CONSIDERATIONS

You never send the `GXFinishSendPage` message yourself.

If you are implementing a nonsupported type of communications connection, you need to perform a total override of the `GXFinishSendPage` message. Otherwise, you need to perform your tasks and then forward the message.

RESULT CODES

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

The default implementation of the `GXFinishSendPage` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `GXImagePage` message is described on page 4-94.

GXBufferData

QuickDraw GX sends the `GXBufferData` message to send data to the device using the standard buffering mechanism. You can override the `GXBufferData` message to provide your own buffering scheme or to change data prior to buffering. Your override of the `GXBufferData` message must match the following formal declaration:

```
OSErr MyBufferData (Ptr data, long length, long bufferOptions);
```

`data`	A pointer to the data to add to the buffer.
`length`	The number of bytes of data to add.
`bufferOptions`	Options for adding the data to the buffer, as shown in Table 4-10.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXBufferData` message in preparation for sending data to a device using the standard buffering mechanism. This message is sent whenever any message handler wishes to buffer the data to be sent to the printing device.

The default implementation of this message uses the default buffering. It buffers the data until all of the buffers are full. You can specify buffering options in the `bufferOptions` parameter. The constants for these options are shown in Table 4-10.

Table 4-10 Options for adding data to the buffer

Constant	Value	Explanation
`gxNoBufferOptions`	`0x00000000`	No options apply.
`gxMakeBufferHex`	`0x00000001`	All data is converted into ASCII hex equivalents: each byte becomes 2 ASCII bytes.
`gxDontSplitBuffer`	`0x00000002`	The data sent in this message cannot be split across buffers. If it is too large to fit into what is left of the current buffer, then add it to the next buffer instead.

When the buffers are completely full, QuickDraw GX sends the `GXDumpBuffer` message to write the data in the buffer to the device and then sends the `GXFreeBuffer` message to open the buffer to write additional data.

SPECIAL CONSIDERATIONS

You can send the `GXBufferData` message to add data to the printer's data stream.

If you are implementing your own buffering scheme, you need to perform a total override of the `GXBufferData` message. Otherwise, you need to modify the data and then forward the message.

RESULT CODES

`gxSegmentLoadFailedErr`	A required code segment could not be found, or there was not enough memory to load it.
`gxPrUserAbortErr`	The user has canceled printing.
`gxGBBufferTooSmallErr`	The buffer is too small to accept the requested data, and you have specified that buffers are not to be split.

The default implementation of the `GXBufferData` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `GXDumpBuffer` message is described on page 4-142.

The `GXFreeBuffer` message is described on page 4-143.

GXWriteData

QuickDraw GX sends the `GXWriteData` message to send data directly to the device, without buffering. You can override the `GXWriteData` message to provide your own I/O mechanism. Your override of the `GXWriteData` message must match the following formal declaration:

```
OSErr MyWriteData (Ptr data, long length);
```

`data`	A pointer to the data to send to device.
`length`	The number of bytes of data.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The `GXWriteData` message sends unbuffered data to a device and waits for I/O to complete.

The default implementation of this message calls the asynchronous I/O functions of the Macintosh system software to send the data to the device. The default implementation then flushes the buffers, sends the resulting data through, and waits for the I/O operation to complete.

You can override this message as part of implementing your own I/O and buffering mechanism or changing how I/O is performed with your device.

SPECIAL CONSIDERATIONS

You can use the `GXWriteData` message yourself to send data to your device without buffering or to force the buffers to be flushed.

If you are implementing your own I/O mechanism, you need to totally override the `GXWriteData` message. If you are using the standard QuickDraw GX buffering and Macintosh I/O system, you must forward this message.

RESULT CODES

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

The default implementation of the `GXWriteData` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

GXDumpBuffer

QuickDraw GX sends the `GXDumpBuffer` message in order to write to the device the data in the buffers. You can override the `GXDumpBuffer` message to process data and buffers in your own way. Your override of the `GXDumpBuffer` message must match the following formal declaration:

```
OSErr MyDumpBuffer (gxPrintingBuffer *aPrintingBuffer);
```

`aPrintingBuffer`
A pointer to the printing buffer structure that defines the buffer to send to the device.

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

DESCRIPTION

QuickDraw GX sends the `GXDumpBuffer` message to write data to the device. It sends the message when any one of the buffers is full and data must be sent.

The default implementation of this message calls the asynchronous I/O functions of the Macintosh system software to send the buffer to the device. The default implementation then sends the requested buffer to the device using PAP, serial, or not-connected communications, depending on what you specified in your printing extension or printer driver resources. You can override this message to process data and buffers in your own way.

SPECIAL CONSIDERATIONS

You can send the `GXDumpBuffer` message yourself if you are implementing your own buffering scheme.

If you are implementing your own I/O mechanism, you need to perform a total override of the `GXDumpBuffer` message. Otherwise, you need to forward this message to the other printing message handlers.

If you override the `GXDumpBuffer` message, you must also override the `GXFreeBuffer` message.

RESULT CODES

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

The default implementation of the `GXDumpBuffer` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `GXFreeBuffer` message is described in the next section.

The printing buffer structure is described on page 4-11.

GXFreeBuffer

QuickDraw GX sends the `GXFreeBuffer` message when waiting for the completion of a buffer that has been sent with the `GXDumpBuffer` message. You can override the `GXFreeBuffer` message if you are processing data and buffers in your own way. Your override of the `GXFreeBuffer` message must match the following formal declaration:

```
OSErr MyFreeBuffer (gxPrintingBufferPtr *aPrintingBuffer);
```

`aPrintingBuffer`
A pointer to the printing buffer structure that defines the buffer to wait for.

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

DESCRIPTION

The default implementation of the `GXFreeBuffer` message waits for processing of the buffer to complete using PAP, serial, or not-connected communications. It calls the asynchronous I/O functions of the Macintosh system software to wait for the buffer to complete transmission to the device. QuickDraw GX sends this message when it needs a buffer and they are all full.

You can override this message if you are processing data and buffers in your own way. You must not return from this message until the I/O completes or terminates with an error.

SPECIAL CONSIDERATIONS

You can send the `GXFreeBuffer` message yourself if you are implementing your own buffering scheme.

If you override the `GXDumpBuffer` message, you must also override the `GXFreeBuffer` message.

If you are implementing your own I/O mechanism, you need to perform a total override of the `GXFreeBuffer` message.

RESULT CODES

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

The default implementation of the `GXFreeBuffer` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `GXDumpBuffer` message is described in the previous section.

The printing buffer structure is described on page 4-11.

GXFinishSendPlane

QuickDraw GX sends the `GXFinishSendPlane` message to mark the end of processing of a plane or "color pass" for raster devices. You can override the `GXFinishSendPlane` message to finish processing of a color plane and send it to your device. Your override of the `GXFinishSendPlane` message must match the following formal declaration:

```
OSErr MyFinishSendPlane (void);
```

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

DESCRIPTION

QuickDraw GX sends this message to mark the end of a color plane for raster devices that require multiple passes for each page. A film recorder might send this message to mark the end of the red, green, and blue color passes.

QuickDraw GX does not have a default implementation of the `GXFinishSendPlane` message.

RESULT CODES

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

SPECIAL CONSIDERATIONS

You can send the `GXFinishSendPlane` message yourself to mark completion of a plane.

You can totally override the `GXFinishSendPlane` message to perform tasks at the completion of a color plane pass.

GXCheckStatus

QuickDraw GX sends the `GXCheckStatus` message to mark a location in the communications process where a status check needs to be performed. You need to override the `GXCheckStatus` message to check the status of your device. Your override of the `GXCheckStatus` message must match the following formal declaration:

```
OSErr MyCheckStatus (Ptr data, long length, short statusType,
                          Signature owner);
```

`data`	A pointer to the data that you want to send with the status.
`length`	The number of bytes in the data.
`statusType`	The application-defined status type.
`owner`	The ID of the message handler for which the `GXCheckStatus` message is intended.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

You can send the `GXCheckStatus` message to mark the point in the imaging-communications process where you can check status information about the device.

The default implementation of this message does nothing. You need to override this message to provide status checking for your device.

The message handler that sends the `GXCheckStatus` message sets the `data`, `length`, and `statusType` parameters. You need to check the `owner` parameter and only process the status check if it matches your signature; if not, you need to forward the message. Most message handlers call the `GXGetDeviceStatus` message in response to the `GXCheckStatus` message.

SPECIAL CONSIDERATIONS

You can send the `GXCheckStatus` message yourself to mark a place where you want to have a status check performed.

If you process the status check in your override, you do not forward the `GXCheckStatus` message. If you do not handle the status check, you must forward the message.

RESULT CODES

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

SEE ALSO

The `GXGetDeviceStatus` message is described in the next section.

GXGetDeviceStatus

QuickDraw GX sends the `GXGetDeviceStatus` message to query the device about its status. You can override the `GXGetDeviceStatus` message to send a query to your own device about its status. Your override of the `GXGetDeviceStatus` message must match the following formal declaration:

```
OSErr MyGetDeviceStatus (Ptr cmdData, long cmdSize,
                         Ptr responseData, long *responseSize,
                         Str255 termination);
```

`cmdData` A pointer to the status query command data.

`cmdSize` The number of bytes in the command data.

`responseData`
: On return, a pointer to the data that contains the device's response.

`responseSize`
: On entry, the number of bytes to read. On return, the number of bytes in the response data.

`termination`
: This is the string that marks the end of the response in the `responseData` parameter. If this is `nil`, you need to read all of the bytes in the response data.

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

DESCRIPTION

Before sending the `GXGetDeviceStatus` message yourself, you need to make sure that all pending I/O with your device is flushed. This can be done by sending a `GXWriteData` message with a `nil` pointer and a length of 0.

The default implementation of this message supports PAP, serial, and not-connected communications by writing a status request and reading back the result from the device.

SPECIAL CONSIDERATIONS

You can send the `GXGetDeviceStatus` message when you want to perform a status check of a device.

If you are implementing a nonsupported type of communications connection, you need to perform a total override of the `GXGetDeviceStatus` message. Otherwise, you need to perform your tasks and then forward the message.

RESULT CODES

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

The default implementation of the `GXGetDeviceStatus` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

SEE ALSO

The `GXWriteData` message is described on page 4-141.

Compatibility Messages

QuickDraw GX sends these messages only when an application makes calls to the Macintosh Printing Manager. QuickDraw GX responds to these messages by mapping the Macintosh Printing Manager behavior into QuickDraw GX actions. If you are writing drivers, these message allow you to customize the behavior of these printing calls, although you can most easily accomplish the basic mapping with the resources that are described in the chapter “Printing Resources” in this book.

If you need to learn about the Macintosh Printing Manager and how these calls were used to implement QuickDraw printing, refer to *Inside Macintosh: Imaging With QuickDraw.*

QuickDraw GX looks for dialog resources (`'DLOG'`, `'DITL'`, and `'dctl'` resources) with specific resource IDs to find the dialog information that is used by some of the compatibilty messages, including the `GXPrStlDialog`, `GXPrJobDialog`, `GXPrStlInit`, `GXPrJobInit`, and `GXPrDlgMain` functions. QuickDraw GX looks for dialog resources with ID = –8192 for style dialog information and with ID = –8191 for job dialog information. If QuickDraw GX does not find dialog resources with these IDs in the driver resources, it uses default versions. The dialog control (`'dctl'`) resource is described in the section “The Dialog Control ('dctl') Resource” beginning on page 6-52 in the chapter “Printing Resources.” The dialog (`'DLOG'`) and dialog item list (`'DITL'`) resources are described in *Inside Macintosh: Macintosh Toolbox Essentials*.

GXPrOpenDoc

QuickDraw GX sends the `GXPrOpenDoc` message when an application that supports the Macintosh Printing Manager calls the `PrOpenDoc` function. You can override the `GXPrOpenDoc` message to customize the handling of the `PrOpenDoc` function. Your override of the `GXPrOpenDoc` message must match the following formal declaration:

```
OSErr MyPrOpenDoc (THPrint aTHPrint, TPPrPort *aTPPrPort);
```

`aTHPrint`	A handle to the print record for this printing operation.
`aTPPrPort`	A pointer to a `TPrPort` record.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The default implementation of the `GXPrOpenDoc` sends the `GXStartJob` message. You can override the `GXPrOpenDoc` message to add any special handling that your printer driver requires at the time that a document is spooled for printing.

SPECIAL CONSIDERATIONS

You never send the `GXPrOpenDoc` message yourself.

You almost always forward the `GXPrOpenDoc` message so that the default implementation can perform its operations.

RESULT CODES

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

SEE ALSO

The `PrOpenDoc` function is described in *Inside Macintosh: Imaging With QuickDraw.*

The `GXStartJob` message is described on page 4-52.

GXPrCloseDoc

QuickDraw GX sends the `GXPrCloseDoc` message when an application that supports the Macintosh Printing Manager calls the `PrCloseDoc` function. You can override the

GXPrCloseDoc message to customize the handling of the PrCloseDoc function. Your override of the GXPrCloseDoc message must match the following formal declaration:

```
OSErr MyPrCloseDoc (TPPrPort aTPPrPort);
```

aTPPrPort	A pointer to a TPPrPort record.
function result	An error code. The value noErr indicates that the operation was successful.

DESCRIPTION

The default implementation of the GXPrCloseDoc message generates the GXFinishJob message. You can override this message to add any special handling that your printer driver requires at the time that a document is finished printing.

SPECIAL CONSIDERATIONS

You never send the GXPrCloseDoc message yourself.

You almost always forward the GXPrCloseDoc message so that the default implementation can perform its operations.

RESULT CODES

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

SEE ALSO

The PrCloseDoc function is described in *Inside Macintosh: Imaging With QuickDraw.*

The GXFinishJob message is described on page 4-54.

GXPrOpenPage

QuickDraw GX sends the GXPrOpenPage message when an application that supports the Macintosh Printing Manager calls the PrOpenPage function. You can override the GXPrOpenPage message to customize the handling of the PrOpenPage function. Your override of the GXPrOpenPage message must match the following formal declaration:

```
OSErr MyPrOpenPage (TPPrPort aTPPrPort, TPRect aTPRect,
                    Point resolution);
```

aTPPrPort	A pointer to a TPPrPort record.

`aTPRect`	The rectangle used as the QuickDraw picture frame for this page.
`resolution`	The resolution to be used in printing the page.

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

DESCRIPTION

The default implementation of the `GXPrOpenPage` message generates the `GXStartPage` message. You can override this message to add any special handling that your printer driver requires at the time that a page is spooled for printing.

SPECIAL CONSIDERATIONS

You never send the `GXPrOpenPage` message yourself.

You almost always forward the `GXPrOpenPage` message so that the default implementation can perform its operations.

RESULT CODES

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

SEE ALSO

The `PrOpenPage` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The `GXStartPage` message is described on page 4-55.

GXPrClosePage

QuickDraw GX sends the `GXPrClosePage` message when an application that supports the Macintosh Printing Manager calls the `PrClosePage` function. You can override the `GXPrClosePage` message to customize the handling of the `PrClosePage` function. Your override of the `GXPrClosePage` message must match the following formal declaration:

```
OSErr GXPrCloseDoc (TPPrPort aTPPrPort);
```

`aTPPrPort` A pointer to a `TPPrPort` record.

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

DESCRIPTION

The default implementation of this message generates the `GXFinishPage` message. You can override this message to add any special handling that your driver requires at the time that a page has finished printing.

SPECIAL CONSIDERATIONS

You never send the `GXPrClosePage` message yourself.

You almost always forward the `GXPrClosePage` message so that the default implementation can perform its operations.

RESULT CODES

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

SEE ALSO

The `PrClosePage` function is described in *Inside Macintosh: Imaging With QuickDraw.*

The `GXFinishPage` message is described on page 4-57.

GXPrintDefault

QuickDraw GX sends the `GXPrintDefault` message when an application that supports the Macintosh Printing Manager calls the `PrintDefault` function. You can override the `GXPrintDefault` message to customize the handling of the `PrintDefault` function. Your override of the `GXPrintDefault` message must match the following formal declaration:

```
OSErr MyPrintDefault (THPrint aTHPrint);
```

`aTHPrint`	A handle to the print record for this printing operation.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The default implementation of this message loads a default `'PREC'` resource. You can override this message to add any special handling of the default print record.

SPECIAL CONSIDERATIONS

You never send the `GXPrintDefault` message yourself.

You almost always forward the `GXPrintDefault` message so that the default implementation can perform its operations.

RESULT CODES

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

SEE ALSO

The `PrintDefault` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The `'PREC'` resource is described on page 6-51 in the chapter “Printing Resources” in this book.

GXPrStlDialog

QuickDraw GX sends the `GXPrStlDialog` message when an application that supports the Macintosh Printing Manager calls the `PrStlDialog` function. You can override the `GXPrStlDialog` message to customize the handling of the `PrStlDialog` function. Your override of the `GXPrStlDialog` message must match the following formal declaration:

```
OSErr MyPrStlDialog (THPrint aTHPrint, Boolean *aBoolean);
```

`aTHPrint`	A handle to the print record for this printing operation.
`aBoolean`	On return, a Boolean value that is `true` if the user confirmed the dialog box and `false` if not.

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

DESCRIPTION

The default implementation of the `GXPrStlDialog` message looks for dialog (`'DLOG'`), item list (`'DITL'`), and dialog control (`'dctl'`) resources that have the ID –8192 in your driver resources. If QuickDraw GX does not find resources with this ID, it uses its own default resources to construct and display a style dialog box. Some of the values displayed in the style dialog box are controlled by the customization (`'cust'`)resource.

You can override the `GXPrStlDialog` message to add any special handling that your printer driver requires with regard to page dimensions and page setup.

SPECIAL CONSIDERATIONS

You never send the `GXPrStlDialog` message yourself.

You almost always forward the `GXPrStlDialog` message so that the default implementation can perform its operations.

RESULT CODES

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

SEE ALSO

The `PrStlDialog` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The dialog control and customization resources are described in the chapter “Printing Resources” in this book.

The dialog and item list resources are described in *Inside Macintosh: Macintosh Toolbox Essentials*.

GXPrJobDialog

QuickDraw GX sends the `GXPrJobDialog` message when an application that supports the Macintosh Printing Manager calls the `PrJobDialog` function. You can override the `GXPrJobDialog` message to customize the handling of the `PrJobDialog` function. Your override of the `GXPrJobDialog` message must match the following formal declaration:

```
OSErr MyPrJobDialog (THPrint aTHPrint, Boolean *aBoolean);
```

`aTHPrint`	A handle to the print record for this printing operation.
`aBoolean`	On return, a Boolean value that is `true` if the user confirmed the dialog box and `false` if not.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The default implementation of the `GXPrJobDialog` message looks for dialog (`'DLOG'`), item list (`'DITL'`), and dialog control (`'dctl'`) resources that have the ID –8191 in your driver resources. If QuickDraw GX does not find resources with this ID, it uses its own default resources to construct and display a Print dialog box. Some of the values displayed in the Print dialog box are controlled by the customization (`'cust'`)resource.

You can override this message to add any special handling that your driver requires with regard to print quality, range of pages, and other properties associated with a print object.

SPECIAL CONSIDERATIONS

You never send the `GXPrJobDialog` message yourself.

You almost always forward the `GXPrJobDialog` message so that the default implementation can perform its operations.

RESULT CODES

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

SEE ALSO

The `PrJobDialog` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The dialog control and customization resources are described in the chapter "Printing Resources" in this book.

The dialog and item list resources are described in *Inside Macintosh: Macintosh Toolbox Essentials*.

GXPrStlInit

QuickDraw GX sends the `GXPrStlInit` message when an application that supports the Macintosh Printing Manager calls the `PrStlInit` function to add controls to the style dialog box that is displayed when the `PrDlgMain` function is called. You can override the `GXPrStlInit` message to customize the handling of the `PrStlInit` function. Your override of the `GXPrStlInit` message must match the following formal declaration:

```
OSErr MyPrStlInit (THPrint aTHPrint, TPPrDlg *aTPPrDlg);
```

`aTHPrint`	A handle to the print record for this printing operation.
`aTPPrDlg`	A pointer to the dialog structure for the style dialog box.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The default implementation of the `GXPrStlInit` message sets up default controls in the style dialog box. You can override this message if you need to add any special handling to the processing of `PrStlInit` calls by your printer driver.

SPECIAL CONSIDERATIONS

You never send the `GXPrStlInit` message yourself.

You almost always forward the `GXPrStlInit` message so that the default implementation can perform its operations.

RESULT CODES

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

SEE ALSO

The `PrStlInit` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The dialog control and customization resources are described in the chapter "Printing Resources" in this book.

The dialog and item list resources are described in *Inside Macintosh: Macintosh Toolbox Essentials*.

GXPrJobInit

QuickDraw GX sends the `GXPrJobInit` message when an application that supports the Macintosh Printing Manager calls the `PrJobInit` function to set up the structure that is used for displaying the Print dialog box when the `PrDlgMain` function is called. You can override the `GXPrJobInit` message to customize the handling of the `GXPrJobInit` function. Your override of the `GXPrJobInit` message must match the following formal declaration:

```
OSErr MyPrJobInit (THPrint aTHPrint, TPPrDlg *aTPPrDlg);
```

`aTHPrint`	A handle to the print record for this printing operation.
`aTPPrDlg`	A pointer to the `TPPrDlg` record for the Print dialog box.

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

DESCRIPTION

The default implementation of the `GXPrJobInit` message sets up default controls in the Print dialog box.You can override this message if you need to add any special handling to the processing of `PrJobInit` calls by your printer driver.

SPECIAL CONSIDERATIONS

You never send the `GXPrJobInit` message yourself.

You almost always forward the `GXPrJobInit` message so that the default implementation can perform its operations.

RESULT CODES

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

SEE ALSO

The `PrJobInit` function is described in *Inside Macintosh: Imaging With QuickDraw*.

GXPrDlgMain

QuickDraw GX sends the `GXPrDlgMain` message when an application that supports the Macintosh Printing Manager calls the `PrDlgMain` function. You can override the `GXPrDlgMain` message to customize the handling of the `PrDlgMain` function. Your override of the `GXPrDlgMain` message must match the following formal declaration:

```
OSErr MyPrDlgMain (THPrint aTHPrint,
                   PDlgInitProcPtr aPDlgInitProcPtr,
                   Boolean *aBoolean);
```

`aTHPrint` A handle to the print record for this printing operation.

`aPDlgInitProcPtr`
A pointer to the procedure used to initialize the Print dialog box.

`aBoolean` On return, a Boolean value that is `true` if the user confirmed the dialog box and `false` if not.

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

DESCRIPTION

The default implementation of the `GXPrDlgMain` message sets up the default controls for the Print dialog box. You can override this message if you need to add any special handling to the processing of `GXPrDlgMain` calls by your printer driver.

SPECIAL CONSIDERATIONS

You never send the `GXPrDlgMain` message yourself.

You almost always forward the `GXPrDlgMain` message so that the default implementation can perform its operations.

RESULT CODES

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

SEE ALSO

The `PrDlgMain` function is described in *Inside Macintosh: Imaging With QuickDraw*.

GXPrValidate

QuickDraw GX sends the `GXPrValidate` message when an application that supports the Macintosh Printing Manager calls the `PrValidate` function. You can override the `GXPrValidate` message to customize the handling of the `PrValidate` function. Your override of the `GXPrValidate` message must match the following formal declaration:

```
OSErr MyPrValidate (THPrint aTHPrint, Boolean *aBoolean);
```

`aTHPrint`	A handle to the print record for this printing operation.
`aBoolean`	On return, a Boolean value that is `true` if the user confirmed the dialog box and `false` if not.
function result	An error code. The value `noErr` indicates that the operation was successful.

IMPORTANT

The default implementation of this message sends a `GXPrintDefault` message if the value of the `devKind` field in the print record is other than 0xA900 or if the value of the `prVersion` field in the print record is other than 8. s

DESCRIPTION

The default implementation of this message validates the print record using the universal print structure. It sends the `GXConvertPrintRecordTo` message, validates the contents of the structure, and then sends the `GXConvertPrintRecordFrom` message.

You can override this message if you need to add any special handling to the processing of `PrValidate` calls by your printer driver.

SPECIAL CONSIDERATIONS

You never send the `GXPrValidate` message yourself.

You only forward the `GXPrValidate` message so that the default implementation can perform its operations if you do not have a custom print-record format.

The customization resource is described on page 6-47 in the chapter "Printing Resources" in this book.

RESULT CODES

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

SEE ALSO

The `PrValidate` function is described in *Inside Macintosh: Imaging With QuickDraw.*

The universal print structure is described in the section "The Universal Print Structure" on page 4-12.

The `GXConvertPrintRecordTo` message is described page 4-161.

The `GXConvertPrintRecordFrom` message is described on page 4-160.

GXPrGeneral

QuickDraw GX sends the `GXPrGeneral` message when an application that supports the Macintosh Printing Manager calls the `PrGeneral` function. You can override the `GXPrGeneral` message to customize the handling of the `PrGeneral` function. Your override of the `GXPrGeneral` message must match the following formal declaration:

```
OSErr MyPrGeneral (Ptr aPtr);
```

`aPtr`	A pointer to the data block used by `GXPrGeneral`.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The default implementation of this message uses the customization (`'cust'`) resource to find the supported resolution for the printer. It also uses any available resolution (`'resl'`) resources. You can override this message if you need to add any special handling to the processing of `GXPrGeneral` calls by your printer driver.

SPECIAL CONSIDERATIONS

You never send the `GXPrGeneral` message yourself.

You almost always forward the `GXPrGeneral` message so that the default implementation can perform its operations.

RESULT CODES

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

SEE ALSO

The `PrGeneral` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The customization and resolution resources are described in the chapter "Printing Resources" in this book.

GXPrJobMerge

QuickDraw GX sends the `GXPrJobMerge` message when an application that supports the Macintosh Printing Manager calls the `PrJobMerge` function. You can override the `GXPrJobMerge` message to customize the handling of the `PrJobMerge` function. Your override of the `GXPrJobMerge` message must match the following formal declaration:

```
OSErr MyPrJobMerge (THPrint aTHPrint1, THPrint aTHPrint2);
```

`aTHPrint1`	A handle to the first print record for this printing operation.
`aTHPrint2`	A handle to the second print record for this printing operation.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The default implementation of this message merges the two print records and calls `GXConvertPrintRecordTo` to add the print information to the job collection so that you can print several documents with a single dialog box. You can override this message if you need to add any special handling to the processing of `GXPrJobMerge` calls for your driver.

SPECIAL CONSIDERATIONS

You never send the `GXPrJobMerge` message yourself.

You almost always forward the `GXPrJobMerge` message so that the default implementation can perform its operations.

RESULT CODES

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

SEE ALSO

The `PrJobMerge` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The `GXConvertPrintRecordTo` function is described in *Inside Macintosh: QuickDraw GX Printing*.

GXConvertPrintRecordFrom

QuickDraw GX sends the `GXConvertPrintRecordFrom` message when an application that supports the Macintosh Printing Manager calls the `GXConvertPrintRecordFrom` function, which converts a print record into a driver-specific format. You can override the `GXConvertPrintRecordFrom` message to customize the handling of the `GXConvertPrintRecordFrom` function. Your override of the `GXConvertPrintRecordFrom` message must match the following formal declaration:

```
OSErr MyConvertPrintRecordFrom (THPrint aTHPrint);
```

`aTHPrint`	A handle to the print record for this printing operation.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The default implementation of the `GXConvertPrintRecordFrom` message does nothing. Your override fills in the print record with information from the job collection.

SPECIAL CONSIDERATIONS

You never send the `GXConvertPrintRecordFrom` message yourself.

You need to totally override the `GXConvertPrintRecordFrom` message to support your Macintosh Printing Manager print-record format.

RESULT CODES

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

SEE ALSO

The `GXConvertPrintRecordFrom` function is described in *Inside Macintosh: QuickDraw GX Printing*.

GXConvertPrintRecordTo

QuickDraw GX sends the `GXConvertPrintRecordTo` message when an application that supports the Macintosh Printing Manager calls the `GXConvertPrintRecordTo` function, which converts a print record into a universal print structure format. You can override the `GXConvertPrintRecordTo` message if you have a printer driver written for the Macintosh Printing Manager whose print format you wish to continue to support. Your override of the `GXConvertPrintRecordTo` message must match the following formal declaration:

```
OSErr MyConvertPrintRecordTo (THPrint aTHPrint);
```

`aTHPrint`	A handle to the print record for this printing operation.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The default implementation of the `GXConvertPrintRecordTo` message does nothing.

SPECIAL CONSIDERATIONS

You never send the `GXConvertPrintRecordTo` message yourself.

You need to totally override the `GXConvertPrintRecordTo` message to support your Macintosh Printing Manager print-record format.

RESULT CODES

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

SEE ALSO

The `GXConvertPrintRecordFrom` function is described in *Inside Macintosh: QuickDraw GX Printing*.

The universal print structure is described on page 4-12.

GXPrintRecordToJob

QuickDraw GX sends the `GXPrintRecordToJob` message when an application that supports the Macintosh Printing Manager calls the `GXPrintRecordToJob` function, which converts a print record into information in the job collection. You can override the `GXPrintRecordToJob` message if you have a printer driver written for the Macintosh Printing Manager with a print record format you wish to continue to support. Your override of the `GXPrintRecordToJob` message must match the following formal declaration:

```
OSErr MyPrintRecordToJob (THPrint aTHPrint, gxJob aJob);
```

`aTHPrint`	A handle to the print record for this printing operation.
`aJob`	The job object.

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

DESCRIPTION

The default implementation of the `GXPrintRecordToJob` message sends the `GXConvertPrintRecordTo` message and then moves all of the universal fields into the job object.

SPECIAL CONSIDERATIONS

You never send the `GXPrintRecordToJob` message yourself.

You can totally or partially override the `GXPrintRecordToJob` message.

RESULT CODES

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

SEE ALSO

The `GXPrintRecordToJob` function is described in *Inside Macintosh: QuickDraw GX Printing*.

The `GXConvertPrintRecordTo` message is described in the previous section.

Finder Dialog Box Messages

The Finder uses Finder dialog box messages when it wants to display information in a dialog box for the user when there are messages or a printing problem in the background. It sends these messages during the Finder interaction phase of printing.

GXWriteStatusToDTPWindow

The Finder sends the `GXWriteStatusToDTPWindow` message when it receives the status from a background printing process and wants to display that status in the the user's desktop printer window. You can override the `GXWriteStatusToDTPWindow` message to determine if you want to handle a specific status message. Your override of the `WriteStatusToDTPWIndow` message must match the following formal declaration:

```
OSErr MyWriteStatusToDTPWindow (gxStatusRecord aStatusRecord,
                                 gxDisplayRecord aDisplayRecord);
```

`aStatusRecord`
: A pointer to the status structure.

`aDisplayRecord`
: On return, a pointer to the information to be displayed.

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

DESCRIPTION

The default implementation of this message turns the status structure into a string using the status (`'stat'`) resource supplied by the driver.

Your override of the `GXWriteStatusToDTPWindow` message needs to examine the `statusOwner` field in the status structure to determine if you need to handle the status. If your override needs to display status information that is more complex than is supported by this standard status handling, you must fill in this structure yourself. If the `statusOwner` field in the status structure does not match your driver's signature, you must forward this message.

SPECIAL CONSIDERATIONS

You never send the `GXWriteStatusToDTPWindow` message yourself.

If you do handle the status, you need to perform a total override of the `GXWriteStatusToDTPWindow` message. Otherwise, you must forward the `GXWriteStatusToDTPWindow` message so that another handler can respond to it.

RESULT CODES

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

SEE ALSO

The status structure is described on page 4-39, and the status resource is described on page 6-19 in the chapter "Printing Resources."

The display structure is described on page 4-41.

GXInitializeStatusAlert

QuickDraw GX sends the `GXInitializeStatusAlert` message when a driver calls the `GXAlertTheUser` function with a status structure containing a status type of `gxUserAlert`. You need to override the `GXInitializeStatusAlert` message if you want to provide custom alerts. Your override of the `GXInitializeStatusAlert` message needs to be declared as follows.

```
OSErr MyInitializeStatusAlert (gxStatusRecord *statRecPtr,
                                      DialogPtr *dPtr);
```

`statRecPtr`
A pointer to the status structure.

`dPtr` A pointer to the dialog box pointer.

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

DESCRIPTION

The default implementation of the `GXInitializeStatusAlert` message is the automatic handling of the standard printing (`'plrt'`) alert boxes.

You do not need to override this message if your alert is one of the standard alerts for which QuickDraw GX provides automatic handling, as described in the chapter "Printer Drivers" in this book. If you want to provide custom alerts, first check the `statusOwner` field of the status structure to make sure that this message is intended for you. If so, perform a total override of this message. You must create a dialog box and return a pointer to it. You can call the `GetNewDialog` function for the specified alert and then perform any necessary dialog box initialization.

SPECIAL CONSIDERATIONS

You never send the `GXInitializeStatusAlert` message yourself.

If the `statusOwner` field of the status structure matches your signature, you perform a total override of the `GXInitializeStatusAlert` message. Otherwise, forward the message so that another message handler can respond to it.

RESULT CODES

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

SEE ALSO

You can find an example of an overide of the `GXInitializeStatusAlert` message in Listing 3-17 on page 3-45 in the chapter "Printer Drivers."

The `GXAlertTheUser` function is described on page 5-18 in the chapter "Printing Functions for Message Overrides."

The `GetNewDialog` function is described in *Inside Macintosh: Macintosh Toolbox Essentials*.

The status structure is described on page 4-39.

GXHandleAlertEvent

You need to override the `GXHandleAlertEvent` message if you are providing custom handling of printing alerts. Your override code of the `GXHandleAlertEvent` message must match the following formal declaration:

```
OSErr MyHandleAlertEvent (gxStatusRecord *aStatusRecord,
 DialogPtr *aDialogPtr, EventRecord *theEvent, short *itemHit);
```

`aStatusRecord`	A pointer to the status structure.
`aDialogPtr`	The handle to the printing alert box.
`theEvent`	A pointer to the event structure containing the alert event.
`itemHit`	The ID of the dialog item that was selected by the user.

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

DESCRIPTION

QuickDraw GX sends the `GXHandleAlertEvent` message when a printer driver calls the `GXAlertTheUser` function with a status structure containing a status type of `gxUserAlert`.

The default implementation of this message handles the standard set of status alerts. If this message is forwarded to the universal driver and if it is not a message for the universal driver, the default implementation assumes that the caller has set up a printing alert box and handles the event accordingly.

You do not need to override this message if your alert is one of the standard alerts for which QuickDraw GX provides automatic handling, as described in the chapter “Printer Drivers” in this book. If you want to provide custom alerts, first check the `statusOwner` field of the status structure to make sure that this message is intended for you. If so, perform a total override of this message.

In your override, you need to check the event. If the status event dismisses the printing alert box, fill in the `dialogResult` field of the status structure with a nonzero value to inform the Finder that the dialog box can be removed.

SPECIAL CONSIDERATIONS

You never send the `GXHandleAlertEvent` message yourself.

If you handle the event, you must totally override the `GXHandleAlertEvent` message. Otherwise, forward the message so that another message handler can process the event.

RESULT CODES

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

SEE ALSO

You can find an example of an override of the `GXHandleAlertEvent` message in Listing 3-18 on page 3-48 in the chapter “Printer Drivers.”

The `GXAlertTheUser` function is described on page 5-18 in the chapter “Printing Functions for Message Overrides.”

The status structure is described on page 4-39.

GXHandleAlertStatus

The Finder or QuickDraw GX may send the `GXHandleAlertStatus` message. You need to override the `GXHandleAlertStatus` message if you want to display status or error dialog boxes that are more complex than those supported by the standard status handling. Your override of the `GXHandleAlertStatus` message must match the following formal declaration:

```
OSErr MyHandleAlertStatus (gxStatusRecord *statRecPtr);
```

`statRecPtr`	A pointer to the status structure.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The Finder sends the `GXHandleAlertStatus` message when it receives the status from a background printing process and wishes to alert the user. QuickDraw GX sends the `GXHandleAlertStatus` message when a printer driver calls the `GXAlertTheUser` function with a status structure containing a status type of `gxUserAttention`.

The default implementation of this message expects that the caller wants to display a printing alert box and performs the actions necessary to do so.

You need to override this message if you want to display status dialog boxes or printing alert boxes that are more complex than are supported by this standard status handling. If you want to override this message, you first check the `statusOwner` field of the status structure to make sure that this message is intended for you. If it is intended for you, you are free to manage the dialog box yourself.

SPECIAL CONSIDERATIONS

You never send the `GXHandleAlertStatus` message yourself.

If you handle the status, perform a total override of the `GXHandleAlertStatus` message. Otherwise, forward the message so that another message handler can process it.

RESULT CODES

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

SEE ALSO

The `GXAlertTheUser` function is described on page 5-18 in the chapter “Printing Functions for Message Overrides” in this book.

The status structure is described on page 4-39.

GXHandleAlertFilter

You need to override the `GXHandleAlertEvent` message if you are providing custom handling of printing alerts. Your override code of the `GXHandleAlertEvent` message must match the following formal declaration:

```
OSErr MyGXHandleAlertFilter (gxJob aJob, gxStatusRecord *aStatus,
                          DialogPtr aDialog, EventRecord *aEventRecord,
                          short *itemHit, Boolean *returnImmed);
```

`ajob`	The job object that is associated with the dialog box.
`aStatus`	A pointer to the status record that describes the alert.
`aDialog`	A pointer to information about the dialog box that is displaying the alert information.
`aEventRecord`	A pointer to information about the event that is being handled.
`itemHit`	A pointer to the ID of the item in the dialog box in which the event occurred.
`returnImmed`	On return, a Boolean value that is `true` if the filter procedure completely handled the event (no further event processing needs to be done) and `false` if the filter procedure did not completely handle the event.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

QuickDraw GX sends the `GXHandleAlertFilter` message when a user-initiated event occurs in a printing alert box.

The default implementation of this message handles events that occur in the the printing alert box. If you have customized the printing alert box, you need to handle those events as necessary.

In your override, you need to check the event. If the status event dismisses the printing alert box, set the `returnImmed` parameter value to `true`.

SPECIAL CONSIDERATIONS

You never send the `GXHandleAlertFilter` message yourself.

If you handle the event, you must totally override the `GXHandleAlertFilter` message. Otherwise, forward the message so that another message handler can process the event.

RESULT CODES

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

Finder Menu Messages

You can use Finder menu messages if you need to provide users with setup options or interaction through the Finder's Printing menu. The Finder sends these messages during printing from the Finder.

GXGetDTPMenuList

The Finder sends the `GXGetDTPMenuList` message when it displays the Printing menu for a particular desktop printer.You can override the `GXGetDTPMenuList` message if you need to add menu items to the Printing menu. Your override of the `GXGetDTPMenuList` message must match the following formal declaration:

```
OSErr MyGetDTPMenuList (MenuHandle aMenu);
```

`aMenu`	A handle to the menu to which you are adding items.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The default implementation of this message provides the initial list of Printing menu items.

You can override this message if you need to add menu items to the Printing menu.You can use the `AppendMenu` function to add your items.

SPECIAL CONSIDERATIONS

You never send the `GXGetDTPMenuList` message yourself.

You must forward the `GXGetDTPMenuList` message. Forward the message first and then add your menu items.

RESULT CODES

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

SEE ALSO

For more information on desktop printers and the Printing menu, see *Inside Macintosh: QuickDraw GX Printing*.

GXDTPMenuSelect

The Finder sends the `GXDTPMenuSelect` message when the user chooses a menu item in the Printing menu for a specific printing device. You must override the `GXDTPMenuSelect` message if you have added items to the Printing menu. Your override of the `GXDTPMenuSelect` message must match the following formal declaration:

```
OSErr MyDTPMenuSelect (short id);
```

`id`	The ID of the menu item chosen from the Printing menu by the user.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

You need to override the `GXDTPMenuSelect` message if you add items to the Printing menu. If the item selected is less than or equal to the number of items you have added, handle that item. If the value is positive, you need to subtract your total number of items from the item value before forwarding the message.

If the value of the `id` parameter is less than or equal to the number of items that you added in your override of the `GXGetDTPMenuList` message, then you need to handle the selection.

If the `id` parameter is greater than the number of items that you added in the `GXGetDTPMenuList` message, then you need to subtract this total from the `id` parameter and forward `GXDTPMenuSelect` to enable the next message handler to utilize the message.

SPECIAL CONSIDERATIONS

You never send the `GXDTPMenuSelect` message yourself.

If you override the `GXDTPMenuSelect` message, you also need to override the `GXGetDTPMenuList` message.

If you handle the chosen menu item, do not forward the `GXDTPMenuSelect` message. Otherwise, you must forward it.

RESULT CODES

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

SEE ALSO

The `GXGetDTPMenuList` message is described in the previous section.

For more information about desktop printers and the Printing menu, see *Inside Macintosh: QuickDraw GX Printing*.

Summary of Printing Messages

Data Types for Printer Drivers

The Print-to-File Structure

```
struct gxPrintDestinationRec {
   Boolean     printToFile;   /* true if printing to file */
   FSSpec      fSpec;         /* print file FSSpec */
   Boolean     includeFonts;  /* true if system fonts included in file */
   Str31       fileFormat;    /* file format name */
};

typedef struct gxPrintDestinationRec gxPrintDestinationRec,
*gxPrintDestinationPtr, **gxPrintDestinationHdl;
```

The Printing Buffer Structure

```
struct gxPrintingBuffer {
   long     size;       /* size of buffer in bytes */
   long     userData;   /* ID assigned by driver or extension for buffer */
   char     data[1];    /* array of data bytes, size bytes long */
};

typedef struct gxPrintingBuffer gxPrintingBuffer;
```

The Page Information Structure

```
struct gxPageInfoRecord {
   long     docPageNum;    /* number of page being printed */
   long     copyNum;       /* copy number being printed */
   Boolean  formatChanged; /* whether format changed from prev */
   Boolean  pageChanged;   /* whether contents changed */
   long     internalUse;   /* private */
};

typedef struct gxPageInfoRecord gxPageInfoRecord;
```

Constants and Data Types for Macintosh Printing Manager Compatibility

The Universal Print Structure

```
struct gxUniversalPrintRecord {
   short    prVersion;        /* print record version */

                              /* prInfo subrecord... */
   short    appDev;           /* device kind, always 0 */
   short    appVRes;          /* application vert resolution */
   short    appHRes;          /* application horiz resolution */
   Rect     appPage;          /* page size, in application resolution */
   Rect     appPaper;         /* paper rectangle [offset from appPage] */

                              /* prStl subrecord... */
   short    devType;          /* device type, always 0 (wDev) */
   short    pageV;            /* page height in 120ths of inch */
   short    pageH;            /* page width in 120ths of inch */
   char     filler;           /* unused */
   char     feed;             /* feed mode */

                              /* prInfoPT subrecord... */
   short       devKind;       /* device kind, always 0xA900 */
   short       devVRes;       /* device vertical resolution */
   short       devHRes;       /* device horizontal resolution */
   Rect        devPage;       /* device page size */

                              /* prXInfo subrecord... */
   short       actualCopies;  /* actual number of copies for this job */
   short       options;       /* options for this device */
   short       reduction;     /* reduction/enlargement factor */
   char        orientation;   /* orientation of paper  */

   char        qualityMode;   /* type of quality mode */
   char        firstTray;     /* type of first feed tray */
   char        remainingTray; /* type of remaining feed tray */

   char        coverPage;     /* type of cover page */
   char        headMotion;    /* type of head motion */
   char        saveFile;      /* type of save file */


   char        userCluster1;  /* unused */
   char        userCluster2;  /* unused */
   char        userCluster3;  /* unused */
```

```
                              /* prJob subrecord... */
   short        firstPage;     /* number of the first page */
   short        lastPage;      /* number of the last page */
   short        copies;        /* # of copies, always 1 */
   char         reserved1;     /* always true, unused */
   char         reserved2;     /* always true, unused */
   PrIdleProcPtr
                pIdleProc;     /* pointer to the idle proc */
   Ptr          pFileName;     /* pointer to the spool filename */
   short        fileVol;       /* spool file volume reference number */
   char         fileVers;      /* file version, must be 0 */
   char         reserved3;     /* always 0 */

   short     printX[19];       /* internal use */
};

typedef struct gxUniversalPrintRecord gxUniversalPrintRecord,
*gxUniversalPrintRecordPtr, **gxUniversalPrintRecordHdl;
```

Feed Mode Options for the Universal Print Structure

```
enum {
   gxAutoFeed     = 0,         /* use automatic paper feeding */
   gxManualFeed   = 1          /* use manual paper feeding */
};
```

Print Options for the Universal Print Structure

```
enum {
   gxPreciseBitmap      = 0x0001,   /* tall adjust(IW), precise bitmap(LW) */
   gxBiggerPages        = 0x0002,   /* no gaps(IW), larger print area(LW) */
   gxGraphicSmoothing   = 0x0004,   /* graphics smoothing for LW */
   gxTextSmoothing      = 0x0008,   /* text smoothing for SC */
   gxFontSubstitution   = 0x0010,   /* font substitution */
   gxInvertPage         = 0x0020,   /* black/white image inversion */
   gxFlipPageHoriz      = 0x0040,   /* flip page image horizontally */
   gxFlipPageVert       = 0x0080,   /* flip page image vertically */
   gxColorMode          = 0x0100,   /* use color printing */
   gxBidirectional      = 0x0200,   /* use bidirectional printing */
   gxUserFlag0          = 0x0400,   /* user flag 0 */
   gxUserFlag1          = 0x0800,   /* user flag 1 */
   gxUserFlag2          = 0x1000,   /* user flag 2 */
   gxReservedFlag0      = 0x2000,   /* reserved flag 0 */
   gxReservedFlag1      = 0x4000,   /* reserved flag 1 */
   gxReservedFlag2      = 0x8000    /* reserved flag 2 */
};
```

Paper Orientation Options for the Universal Print Structure

```
enum {
   gxPortraitOrientation      = 0,  /* use portrait orientation  */
   gxLandscapeOrientation     = 1,  /* use landscape orientation */
   gxAltPortraitOrientation   = 2,  /* use rotated portrait orientation  */
   gxAltLandscapeOrientation  = 3   /* use rotated landscape orientation */
};
```

Print Quality Modes for the Universal Print Structure

```
enum {
   gxBestQuality              = 0,     /* use best-quality printing */
   gxFasterQuality            = 1,     /* use medium-quality printing */
   gxDraftQuality             = 2      /* use draft-quality printing */
};
```

Paper Tray Selections for the Universal Print Structure

```
enum {
   gxFirstTray    = 0,         /* use first paper-feed tray */
   gxSecondTray   = 1,         /* use second paper-feed tray */
   gxThirdTray    = 2          /* use third paper-feed tray */
};
```

Cover Page Options for the Universal Print Structure

```
enum {
   gxNoCoverPage     = 0,         /* don’t print a cover page */
   gxFirstPageCover  = 1,         /* print cover page before first page */
   gxLastPageCover   = 2          /* print cover page after last page */
};
```

Print-Head Motions for the Universal Print Structure

```
enum {
   gxUnidirectionalMotion  = 0,  /* use one-directional print-head motion */
   gxBidirectionalMotion   = 1   /* print in both directions */
};
```

File Save Types for the Universal Print Structure

```
enum {
   gxNoFile          = 0,        /* not saving print job to file */
   gxPostScriptFile  = 1         /* save print job to PostScript file */
};
```

Constants and Data Types for the Raster Imaging System

Raster Offscreen Structure

```
struct gxOffscreenRec {
   short        numberOfPlanes;      /* the number of planes */
   Handle       offscreenStorage;    /* handle to the image data */
   gxOffscreenPlaneRec
                thePlanes[1];        /* array of planes to draw in */
};

typedef struct gxOffscreenRec gxOffscreenRec, *gxOffscreenPtr,
**gxOffscreenHdl;
```

Raster Offscreen Plane Structure

```
struct gxOffscreenPlaneRec {
   gxViewPort     theViewPort;       /* view port for offscreen plane */
   gxViewDevice   theDevice;         /* view device for offscreen plane */
   gxViewGroup    theViewGroup;      /* view group for offscreen plane */
   gxShape        theBitmap;         /* shape object for offscreen bitmap */
   gxBitMap       theBits;           /* actual bitmap data for this area */
};

typedef struct gxOffscreenPlaneRec gxOffscreenPlaneRec;
```

Raster Offscreen Setup Structure

```
struct gxOffscreenSetupRec {
   short       width;                /* width in pixels of raster */
   short       minHeight;            /* minimum height in pixels */
   short       maxHeight;            /* maximum height in pixels */
   Fixed       ramPercentage;        /* maximum % of RAM to use for raster */
   long        ramSlop;              /* amount of RAM to leave free */
   short       depth;                /* depth in bits of each plane */
   gxMapping   vpMapping;            /* mapping for offscreen view ports */
   gxMapping   vdMapping;            /* mapping for offscreen view devices */
   short        planes;              /* number of planes to allocate */
   gxPlaneSetupRec
                planeSetup[4];       /* parameters for each plane */
};

typedef struct gxOffscreenSetupRec gxOffscreenSetupRec;
```

Raster Offscreen Plane Setup Structure

```
struct gxPlaneSetupRec {
   gxRasterPlaneOptions
                   planeOptions;     /* options for this plane */
   gxHalftone      planeHalftone;    /* halftone values for this plane */
   gxColorSpace    planeSpace;       /* color space for this plane */
   gxColorSet      planeSet;         /* color set for this plane */
   gxColorProfile planeProfile;      /* color profile for this plane */
};

typedef struct gxPlaneSetupRec gxPlaneSetupRec;
```

Raster Plane Options

```
enum {
   /* default value: driver allocates bits & creates halftone values */
   gxDefaultOffscreen      = 0x00000000,,
   /* driver does not call the GXSetViewPortHalftone function */
   gxDontSetHalftone       = 0x00000001,
   /* driver calls GXSetViewPortDither function with gxDotType dithering */
   gxDotTypeIsDitherLevel  = 0x00000002
};

typedef long gxRasterPlaneOptions;
```

Raster Package Bitmap Structure

```
struct gxRasterPackageBitmapRec {
   gxBitmap     *bitmapToPackage; /* pointer to bitmap containing the data */
   unsigned short startRaster;   /* raster where packaging begins */
   unsigned short colorBand;     /* color pass of this package */
   Boolean        isBandDirty;   /* whether there are any non-white */
                                 /*  bits in this band */
   Rect           dirtyRect;     /* area containing nonwhite bits*/
};

typedef struct gxRasterPackageBitmapRec gxRasterPackageBitmapRec;
```

Raster Imaging System Structure

```
struct gxRasterImageDataRec {
   /* setup values */
   gxRasterRenderOptions      renderOptions;    /* rendering options */
   Fixed                      hImageRes;        /* horizontal resolution */
   Fixed                      vImageRes;        /* vertical resolution */
   short                      minBandSize; /* minimum band size in pixels */
   short                      maxBandSize; /* maximum band size in pixels */
   gxRectangle                pageSize;         /* size of page in pixels */
   short                      currentYPos;   /* position on page */
   gxRasterPackageRec         packagingInfo; /* raster package structure */
   Boolean                    optionsValid;  /* true if options specified */
   gxRasterPackageControlsRec packageControls; /* raster package controls */
   gxOffscreenSetupRec        theSetup;      /* offscreen setup info */
};

typedef struct gxRasterImageDataRec gxRasterImageDataRec,
*gxRasterImageDataPtr, **gxRasterImageDataHdl;
```

Raster Render Options

```
enum {
   gxDefaultRaster   = 0x00000000,  /* default options */
   gxDontResolveTransferModes
                     = 0x00000001,  /* 0=resolve xfer modes, 1=don’t */
   gxRenderInReverse = 0x00000002,  /* traverse image in reverse */
   gxOnePlaneAtATime = 0x00000004,  /* render each plane separately */
   gxSendAllBands    = 0x00000008   /* send even empty bands */
};

typedef long gxRasterRenderOptions;
```

Raster Package Structure

```
struct gxRasterPackageRec {
   Ptr      bufferSize;       /* buffer size of packaging */
   short    colorPasses;      /* number of color passes */
   short    headHeight;       /* height of print head in pixels */
   short    numberPasses;     /* number of passes per headheight */
   short    passOffset;       /* offset between passes, in pixels */
   gxRasterPackageOptions
            packageOptions;   /* packaging options */
};
```

```
typedef struct gxRasterPackageRec gxRasterPackageRec, *gxRasterPackagePtr,
**gxRasterPackageHdl;
```

Raster Package Options

```
enum {      /* bit fields in gxRasterPackageOptions */
   gxSendAllColors   = 0x00000001,  /* send even clean bands */
   gxInterlaceColor  = 0x00000002,  /* ribbon contamination */
   gxOverlayColor    = 0x00000004,  /* no ribbon problem */
   gxUseColor        = (gxInterlaceColor|gxOverlayColor)
};

typedef long gxRasterPackageOptions;
```

Constants and Data Types for the Vector Imaging System

Vector Haltone Structure

```
struct gxVHalftoneRec {
   gxColorSpace        halftoneSpace;    /* color space for device */
   gxHalftoneCompRec   halftoneComps[4]; /* array of halftone structures */
   long                penIndexForBW;    /* pen index for drawing 1-bit */
                                         /*  deep black&white bitmaps */
};

typedef struct gxVHalftoneRec gxVHalftoneRec;
```

Vector Halftone Component Structure

```
struct gxVHalftoneCompRec {
   Fixed    angle;      /* angle to halftone at—this value must
                             be 0, 45, 90, or 135 */
   long     penIndex;   /* index of the pen to use for this component */
};

typedef struct gxVHalftoneCompRec gxVHalftoneCompRec;
```

Vector Shape Structure

```
struct gxVectorShapeDataRec {
   gxVectorShapeOptions
         shapeOptions;  /* options to control shape handling */
   long  maxPolyPoints; /* max polygon points device can support */
   Fixed shapeError;    /* the allowed deviation from original shape */
   Fixed textSize;      /* delimits filled from outlined text */
   Fixed frameSize;     /* delimits filled from stroked shapes */
};

typedef struct gxVectorShapeDataRec gxVectorShapeDataRec;
```

Vector Shape Options

```
enum {
   gxUnidirectionalFill    = 0x00000001,  /* driver needs to generate scan
                                               lines in 1 direction only */
   gxAlsoOutlineFilledShape= 0x00000002   /* driver also needs to draw the
                                               outlines of filled shapes */
};

typedef long gxVectorShapeOptions;
```

Vector Imaging System Structure

```
struct gxVectorImageDataRec {/* vector imaging system structure */
   gxVectorRenderOptions   renderOptions; /* options for vector rendering */
   Fixed                   devRes;        /* resolution of device */
   gxTransform             devTransform;  /* transform object for device */
   gxColorSet              clrSet;        /* set of colors on device */
   gxColor                 bgColor;       /* background color */
   gxVHaltoneRec           halftoneInfo;  /* halftone info for color */
   gxPenTableHdl           hPenTable;     /* complete list of device pens */
   gxRectangle             pageRect;      /* dimensions of the page */
   gxVectorShapeDataRec    shapeData;     /* info on how to render shapes */
};

typedef struct gxVectorImageDataRec gxVectorImageDataRec,
*gxVectorImageDataPtr, **gxVectorImageDataHdl;
```

Vector Render Options

```
enum {
   gxColorSort    = 0x00000001,     /* must use this for pen plotters */
   gxATransferMode= 0x00000002,     /* driver needs to resolve xfer modes */
   gxNoOverlap    = 0x00000004,     /* driver needs to produce
                                        non-overlapping output */
   gxAColorBitmap = 0x00000008,     /* driver needs to produce color bitmap
                                        output */
   gxSortbyPenPos = 0x00000010,     /* driver needs to draw shapes in same
```

```
                                          order as pens in pen table */
   gxPenLessPlotter=0x00000020,     /* output device is raster printer or
                                          plotter */
   gxCutterPlotter= 0x00000040,     /* output device has a cutter */
   gxNoBackGround = 0x00000080      /* shapes that map to background color
                                          are not sent to the device */
};

typedef long gxVectorRenderOptions;
```

Vector Pen Table Structure

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

typedef struct gxPenTable gxPenTable, *gxPenTablePtr, **gxPenTableHdl;
```

Vector Pen Table Entry Structure

```
struct gxPenTableEntry {
   Str31       penName;         /* name of the pen */
   gxColor     penColor;        /* color that’s part of the color set */
   Fixed       penThickness;    /* size of the pen */
   short       penUnits;        /* the units in which pen size is defined */
   short       penPosition;     /* pen position in the carousel; if pen is
                                     not loaded, value is kPenNotLoaded (-1) */

};

typedef struct gxPenTableEntry gxPenTableEntry;
```

Vector Pen Units

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

Constants and Data Types for the PostScript Imaging System

PostScript Imaging System Structure

```
struct gxPostScriptImageDataRec {
   short          languageLevel; /* PostScript language level */
   gxColorSpace   devCSpace;     /* the printer's color space */
   gxColorProfile devCProfile;   /* the printer's color profile */
   gxPostScriptRenderOptions
                  renderOptions; /* rendering options */
   long           pathLimit;     /* maximum number of points in a path */
   short          gsaveLimit;    /* maximum gsaves allowed */
   short          opStackLimit;  /* maximum number of objects in stack */
   scalerStreamFontFlag
                  fontType;      /* stream type to use when downloading a
                                      font */
   long           printerVM;     /* the amount of printer memory */
   long           reserved0;     /* unused */
};

typedef struct gxPostScriptImageDataRec gxPostScriptImageDataRec,
*gxPostScriptImageDataPtr, **gxPostScriptImageDataHandle;
```

PostScript Render Options

```
enum {
   /* driver needs to convert binary data to 7-bit ASCII values */
   gxNeedsAsciiOption           = 0x00000001,
   /* driver needs to issue PostScript comments */
   gxNeedsCommentsOption        = 0x00000002,
   /* driver needs to calculate bounding box values */
   gxBoundingBoxesOption        = 0x00000004,
   /* driver needs to generate non-device-specific Postscript output */
   gxPortablePostScriptOption   = 0x00000008,
   /* driver needs to use Level 2 device-independent color when
       printing to a Level 2 output device */
   gxUseLevel2ColorOption       = 0x00000080
};

typedef long gxPostScriptRenderOptions;
```

PostScript Glyphs Structure

```
struct gxPrinterGlyphsRec {
   gxFont         theFont;    /* the font */
   long           nGlyphs;    /* how many glyphs it contains */
   gxFontPlatform platform;   /* the platform of the font */
   gxFontScript   script;     /* the script code of the font */
   gxFontLanguage language;   /* the language code of the font */
   long           vmUsage;    /* amount of printer memory needed for font */
   unsigned long  glyphBits[1];/* the glyph data */
};

typedef struct gxPrinterGlyphsRec gxPrinterGlyphsRec;
```

PostScript Procedure Set List Structure

```
struct gxProcSetListRec {
   Signature   clientid;      /* unique client ID */
   OSType      controlType;   /* resource type of procedure set */
   short       controlid;     /* resource ID of procedure set */
   OSType      dataType;      /* data type of procedure set */
   long        reserved0;
};

typedef struct gxProcSetListRec gxProcSetListRec, *gxProcSetListPtr,
**gxProcSetListHdl;
```

PostScript Query Results

```
enum {
   gxPrinterOK          = 0,     /* printer is initialized and ok */
   gxInitializePrinter  = 1,     /* printer needs initialization */
   gxFilePrinting       = 2,     /* printing to file */
   gxResetPrinter       = 128    /* printer needs resetting */
};
```

User Interface Constants and Data Types

The Panel Information Structure

```
struct gxPanelInfoRecord {
   gxPanelEvent panelEvt;  /* the event */
   short panelResId;       /* resource ID of current 'ppnl' res */
   DialogPtr pDlg;         /* pointer to dialog */
   EventRecord *theEvent;  /* pointer to event */
   short itemHit;          /* actual item number of event */
   short itemCount;        /* number of items before your items */
   short evtAction;        /* the action that will occur after
                               this event is processed */
   short errorStringId;    /* 'STR ' ID of error string */
   gxFormat theFormat;     /* the current format */
   void *refCon;           /* refCon from gxPanelSetupRecord */
};

typedef struct gxPanelInfoRecord gxPanelInfoRecord;
```

Panel Events

```
enum {
   gxPanelNoEvt      = (gxPanelEvent) 0,     /* no event */
   gxPanelOpenEvt    = (gxPanelEvent) 1,     /* panel is about to open */
   gxPanelCloseEvt   = (gxPanelEvent) 2,     /* panel is about to close */
   gxPanelHitEvt     = (gxPanelEvent) 3,     /* user has selected item */
   gxPanelActivateEvt= (gxPanelEvent) 4,     /* panel has been activated */
   gxPanelDeactivateEvt= (gxPanelEvent) 5,   /* panel has been deactivated */
   gxPanelIconFocusEvt= (gxPanelEvent) 6,    /* focus has changed to icons */
   gxPanelPanelFocusEvt= (gxPanelEvent) 7,   /* focus has changed to panel */
   gxPanelFilterEvt  = (gxPanelEvent) 8,     /* panel event needs to be
                                                 filtered */
   gxPanelCancelEvt  = (gxPanelEvent) 9,     /* panel has been canceled */
   gxPanelConfirmEvt = (gxPanelEvent) 10,    /* panel has been confirmed */
   gxPanelDialogEvt  = (gxPanelEvent) 11,    /* panel event to be handled
                                                 by the dialog box handler */
   gxPanelOtherEvt   = (gxPanelEvent) 12,    /* an OS event has occurred
                                                 in the panel */
   gxPanelUserWillConfirmEvt
                     = (gxPanelEvent) 13     /* user has selected confirm */
};

typedef long gxPanelEvent;
```

Panel Responses

```
enum {
   gxPanelNoResult            = 0,  /* no result from panel */
   gxPanelCancelConfirmation  = 1,  /* user confirmed panel, but panel
                                        handler discovered an error */
};

typedef long gxPanelResult;
```

Panel Event Actions

```
enum {
   gxOtherAction      = 0,       /* current item doesnt change after event */
   gxClosePanelAction = 1,       /* panel is closed after event */
   gxCancelDialogAction = 2,     /* dialog box is canceled after event */
   gxConfirmDialogAction= 3      /* dialog box is confirmed after event */
};
```

Parse Range Results

```
enum {
   gxRangeNotParsed  = (gxParsePageRangeResult) 0,    /* not parsed yet */
   gxRangeParsed     = (gxParsePageRangeResult) 1,    /* successful parse */
   gxRangeBadFromValue= (gxParsePageRangeResult) 2,   /* the “from page” */
                                                      /*  value is invalid */
   gxRangeBadToValue = (gxParsePageRangeResult) 3     /* the “to page” */
                                                      /*  value is invalid */
};

typedef short gxParsePageRangeResult;
```

The 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 */
};

typedef struct gxStatusRecord gxStatusRecord;
```

The Manual Feed Structure

```
struct gxManualFeedRecord {
   Boolean  canAutoFeed;   /* whether driver can switch to auto feed */
   Str31    paperTypeName; /* name of paperType to feed */
};

typedef struct gxManualFeedRecord gxManualFeedRecord;
```

The Display Structure

```
struct gxDisplayRecord {   /* for GXWriteStatusToDTPWindow message */
   Boolean  useText;       /* true if displaying text, false for picture */
   Handle   hPicture;      /* handle for the picture to display
                               (unimplemented & reserved for future use) */
   Str255   theText;       /* the text to display */
};

typedef struct gxDisplayRecord gxDisplayRecord;
```

Printing Messages

Storage Messages

```
OSErr GXInitialize           (void);
OSErr GXShutDown             (void);
OSErr GXFetchTaggedData      (ResType aResType, short id,
                              Handle *aHandle, Signature owner);
```

Print Object Messages

```
OSErr GXDefaultJob           (void);
OSErr GXDefaultFormat        (gxFormat aFormat);
OSErr GXDefaultPaperType     (gxPaperType aPaperType);
OSErr GXDefaultPrinter       (gxPrinter aPrinter);
OSErr GXDefaultDesktopPrinter
                             (Str31 dtpName);
```

Application Messages

```
OSErr GXStartJob             (StringPtr docName, long pageCount);
void GXCleanupStartJob       (void);
OSErr GXFinishJob            (void);
OSErr GXJobIdle              (void);
OSErr GXStartPage           (gxFormat aFormat, long numViewPorts,
                             gxViewPort *viewPortList);
void GXCleanupStartPage     (void);
OSErr GXFinishPage          (void);
void GXPrintPage            (gxFormat aFormat, gxShape aPage);
void GXJobFormatModeQuery   (gxQueryType aQueryType,
                             void *srcData, void *dstData);
OSErr GXParsePageRange      (StringPtr fromString, StringPtr toString,
                             gxParsePageRangeResult *result);
```

Paper-Handling Messages

```
OSErr GXDoesPaperFit        (gxTrayIndex whichTray, gxPaperType paper,
                             Boolean *fits);
```

Color Profile Messages

```
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);
```

Spooling Messages

```
OSErr GXCreateSpoolFile     (FSSpecPtr aFSSpecPtr,
                             long createOptions, gxSpoolFile *aSpoolFile);
OSErr GXSpoolPage           (gxSpoolFile aSpoolFile, gxFormat aFormat,
                             gxShape aShape);
OSErr GXSpoolData           (gxSpoolFile aSpoolFile, Ptr data,
                             long *length);
OSErr GXSpoolResource       (gxSpoolFile aSpoolFile, Handle aResource,
                             ResType aType, short id);
OSErr GXCompleteSpoolFile   (gxSpoolFile aSpoolFile);
```

Despooling Messages

```
OSErr GXCountPages          (gxSpoolFile aSpoolFile, long *numPages);
OSErr GXDespoolPage          (gxSpoolFile aSpoolFile,
                              long pageNum, gxFormat aFormat,
                              gxShape *aShape, Boolean *formatChanged);
OSErr GXDespoolData          (gxSpoolFile aSpoolFile,
                              Ptr data, long *length);
OSErr GXDespoolResource      (gxSpoolFile aSpoolFile,
                              ResType aType, short id,
                              Handle *aResource);
OSErr GXExamineSpoolFile     (gxSpoolFile aSpoolFile);
OSErr GXCloseSpoolFile       (gxSpoolFile aSpoolFile,
                              long closeOptions);
```

Dialog Box Messages

```
OSErr GXPrintingEvent        (EventRecord *anEventRecord,
                              Boolean filterEvent);
OSErr GXJobDefaultFormatDialog
                             (gxDialogResult *aDialogResult);
OSErr GXFormatDialog         (gxFormat aFormat, StringPtr title,
                              gxDialogResult *aDialogResult);
OSErr GXJobPrintDialog       (gxDialogResult *aDialogResult);
OSErr GXHandlePanelEvent     (gxPanelInfoRecord *aPanelInfoRecord
                              gxPanelResult *panelResult);
OSErr GXFilterPanelEvent     (gxPanelInfoRecord *aPanelInfoRecord;
                              Boolean *returnImmed);
```

Universal Imaging Messages

```
OSErr GXJobStatus            (gxStatusRecord *aStatusRecord);
OSErr GXCaptureOutputDevice  (Boolean capture);
OSErr GXImageJob             (gxSpoolFile aSpoolFile,
                              long *closeOptions);
OSErr GXCreateImageFile      (FSSpecPtr aFSSpecPtr,
                              long imageFileOptions, long *fileReference);
OSErr GXSetupImageData       (void *imageData);
OSErr GXImageDocument        (gxSpoolFile aSpoolFile, void *imageData);
OSErr GXImagePage            (gxSpoolFile aSpoolFile, long pageNumber,
                              gxFormat aFormat, void *imageData);
OSErr GXRenderPage           (gxFormat aFormat,gxShape aShape,
                              gxPageInfoRecord *aPageInfoRecord,
                              void *imageData);
```

Raster Imaging Messages

```
OSErr GXRasterDataIn        (gxOffscreenHdl offScreen,
                             gxRectangle *bandRectangle,
                             gxRectangle *dirtyRectangle);
OSErr GXRasterLineFeed      (short *lineFeedSize,
                             Ptr buffer, unsigned long *bufferPos,
                             gxRasterImageDataHdl imageData);
OSErr GXRasterPackageBitmap (gxRasterPackageBitmapRec *whattoPackage,
                             Ptr buffer, unsigned long *bufferPos,
                             gxRasterImageDataHdl imageData);
```

PostScript Imaging Messages

```
OSErr GXPostScriptQueryPrinter
                            (long *queryResult);
OSErr GXPostScriptInitializePrinter
                            (void);
OSErr GXPostScriptResetPrinter
                            (void);
OSErr GXPostScriptExitServer(void);
OSErr GXPostScriptGetStatusText
                            (Handle statusTextHdl);
OSErr GXPostScriptGetPrinterText
                            (Handle printerTextHdl);
OSErr GXPostScriptScanStatusText
                            (Handle statusTextHdl);
OSErr GXPostScriptScanPrinterText
                            (Handle printerTextHdl);
OSErr GXPostScriptGetDocumentProcSetList
                            (gxProcSetListHdl procSetListHdl,
                             gxPostScriptImageDataHandle hImageData);
OSErr GXPostScriptDownloadProcSetList
                            (gxProcSetListHdl procSetListHdl,
                             gxPostScriptImageDataHandle hImageData);
OSErr GXPostScriptGetPrinterGlyphsInformation
                            (gxPrinterGlyphsRec *glyphPtr);
OSErr GXPostScriptStreamFont(gxFont fontRef,gxScalerStream *stream);
OSErr GXPostScriptDoDocumentHeader
                            (gxPostScriptImageDataHandle hImageData);
OSErr GXPostScriptDoDocumentSetup
                            (gxPostScriptImageDataHandle hImageData);
OSErr GXPostScriptDoDocumentTrailer
                            (gxPostScriptImageDataHandle hImageData);
```

```
OSErr GXPostScriptDoPageSetup
                            (gxFormat aFormat, long pageIndex,
                             gxPostScriptImageDataHandle hImageData);
OSErr GXPostScriptSelectPaperType
                            (gxPaperType aPaperType,long pageIndex,
                             gxPostScriptImageDataHandle hImageData);
OSErr GXPostScriptDoPageTrailer
                            (gxPostScriptImageDataHandle hImageData);
OSErr GXPostScriptEjectPage (gxPaperType aPaperType, long pageIndex,
                             long copiesCount, short erasePage,
                             gxPostScriptImageDataHandle hImageData);
OSErr GXPostScriptProcessShape
                            (gxShape aShape, long count,
                             gxTransform list[]);
```

Vector Imaging Messages

```
OSErr GXVectorPackageShape  (gxShape aShape, long penIndex);
OSErr GXVectorLoadPens      (gxPenTableHdl penTable,
                             long *shapeCounts, Boolean *penTableChanged);
OSErr GXVectorVectorizeShape(gxShape aShape, long penIndex,
                             gxVectorShapeDataRec *aVectorShapeDataRec);
```

Device Communications Messages

```
OSErr GXOpenConnection      (void);
OSErr GXOpenConnectionRetry (ResType commType, void *commData,
                             Boolean *pRetry, OSErr saveErr);
void GXCleanupOpenConnection(void);
OSErr GXCloseConnection     (void);
OSErr GXStartSendPage       (gxFormat pageFormat);
void GXCleanupStartSendPage (void);
OSErr GXFinishSendPage      (void);
OSErr GXBufferData          (Ptr data, long length, long bufferOptions);
OSErr GXWriteData           (Ptr data, long length);
OSErr GXDumpBuffer          (gxPrintingBuffer *aPrintingBuffer);
OSErr GXFreeBuffer          (gxPrintingBufferPtr *aPrintingBuffer);
OSErr GXFinishSendPlane     (void);
OSErr GXCheckStatus         (Ptr data, long length, short statusType,
                             Signature owner);
OSErr GXGetDeviceStatus     (Ptr cmdData, long cmdSize,
                             Ptr responseData, long *responseSize,
                             Str255 termination);
```

Compatibility Messages

```
OSErr GXPrOpenDoc           (THPrint aTHPrint, TPPrPort *aTPPrPort);
OSErr GXPrCloseDoc          (TPPrPort aTPPrPort);
OSErr GXPrOpenPage          (TPPrPort aTPPrPort, TPRect aTPRect,
                             Point resolution);
OSErr GXPrClosePage         (TPPrPort aTPPrPort);
OSErr GXPrintDefault        (THPrint aTHPrint);
OSErr GXPrStlDialog         (THPrint aTHPrint, Boolean *aBoolean);
OSErr GXPrJobDialog         (THPrint aTHPrint, Boolean *aBoolean);
OSErr GXPrStlInit           (THPrint aTHPrint, TPPrDlg *aTPPrDlg);
OSErr GXPrJobInit           (THPrint aTHPrint, TPPrDlg *aTPPrDlg);
OSErr GXPrDlgMain           (THPrint aTHPrint,
                             PDlgInitProcPtr aPDlgInitProcPtr,
                             Boolean *aBoolean);
OSErr GXPrValidate          (THPrint aTHPrint, Boolean *aBoolean);
OSErr GXPrGeneral           (Ptr aPtr);
OSErr GXPrJobMerge          (THPrint aTHPrint1, THPrint aTHPrint2);
OSErr GXConvertPrintRecordFrom
                            (THPrint aTHPrint);
OSErr GXConvertPrintRecordTo(THPrint aTHPrint);
OSErr GXPrintRecordToJob    (THPrint aTHPrint, gxJob aJob);
```

Finder Dialog Box Messages

```
OSErr GXWriteStatusToDTPWindow
                            (gxStatusRecord aStatusRecord,
                             gxDisplayRecord aDisplayRecord);
OSErr GXInitializeStatusAlert
                            (gxStatusRecord *statRecPtr,
                             DialogPtr *dPtr);
OSErr GXHandleAlertEvent    (gxStatusRecord *aStatusRecord,
                             DialogPtr *aDialogPtr, EventRecord *theEvent,
                             short *itemHit);
OSErr GXHandleAlertStatus   (gxStatusRecord *statRecPtr);
OSErr GXHandleAlertFilter   (gxJob ajob, gxStatusRecord *aStatus,
                             DialogPtr aDialog, EventRecord *aEventRecord,
                             short *itemNum, Boolean *returnImmed);
```

Finder Menu Messages

```
OSErr GXGetDTPMenuList      (MenuHandle aMenu);
OSErr GXDTPMenuSelect       (short id);
```

C H A P T E R 5

Printing Functions for Message Overrides

Contents

This chapter describes the printing functions that you can call only from within your overrides of the printing messages. These functions perform a variety of useful operations. You need to call many of these functions when implementing a QuickDraw GX printer driver, and you can call many of them when writing a QuickDraw GX printing extension.

Before reading this chapter, you need to understand how to write QuickDraw GX printing extensions and printer drivers, as described in the chapters "Printing Extensions" and "Printer Drivers" in this book. You also need to understand the QuickDraw GX printing architecture, which is described in *Inside Macintosh: QuickDraw GX Printing*.

This chapter begins with a brief overview of the QuickDraw GX printing functions and then describes how to use them to perform tasks related to implementing a printing extension or printer driver. The section "Printing Functions Reference" beginning on page 5-13, provides a description for each of the functions and for each of the data types that is used with the functions.

About the Printing Functions

You can only call the printing functions from within your implementations of printing message overrides. You call these functions to perform certain tasks while overriding a printing message. These functions are implemented by QuickDraw GX, just like the QuickDraw GX functions that you use to display shapes.

s **WARNING**
You can only call the functions that are described in this chapter from within an override of a QuickDraw GX printing message. If you call one of these functions in any other context, your program will crash. s

In its implementation of some of these functions, QuickDraw GX sends a corresponding printing message to the handlers in the message chain. For example, you can call the `GXJobIdle` function, the implementation of which sends the `GXJobIdle` message to the top of the message chain. Each message handler can then respond to the message as required.

The printing cleanup functions work in a slighly different manner. The QuickDraw GX implementation of each of the printing cleanup functions sends a corresponding printing message; however, the message is sent to the next handler in the chain, not to the top of the chain. For example, when you call the `GXCleanupOpenConnection` function, QuickDraw GX sends the `GXCleanupOpenConnection` message to the handler after your printing extension or printer driver in the message chain. QuickDraw GX implements the printing cleanup functions in this way to ensure that cleanup messages are sent to the same handlers in the same order as were the original printing messages.

Using the Printing Functions

You use the printing functions to perform actions from within your overrides of printing messages. You generally use these functions from within message overrides in a printer driver, though you can also use them in printing extension code.

You use the printing functions to

- display status information and printing alert boxes to the user
- manage which paper type is assigned to each paper tray on a printer
- store and access information in a desktop printer
- communicate which imaging options your driver provides to applications
- access data that your printing extension or printer driver needs from the job collection and Macintosh system software
- clean up after an error occurs during the processing of certain printing messages

This section provides general information about using the functions for each of these tasks.

Displaying Status Information and the Printing Alert Boxes

QuickDraw GX defines two types of user feedback that you can provide from your printing extension or printer driver: status information and printing alert boxes. You display status information in a user's desktop printer window when the information does not require any user response. You display printing alert boxes when you need the user to respond prior to continuing the printing process.

You use the `GXReportStatus` function to display status information to the user in the desktop printer window. You use the `GXAlertTheUser` function to display printing alert boxes to the user. Both of these functions are described in the section "Displaying Status Information and the Printing Alert Boxes" beginning on page 3-41 in the chapter "Printer Drivers."

QuickDraw GX also provides the printing alert (`'plrt'`) resource for defining printing alert boxes. The use of printing alert boxes is also described in the section "Displaying Status Information and the Printing Alert Boxes" beginning on page 3-41 in the chapter "Printer Drivers." You use the `GXGetPrintingAlert` function to display a printing alert box.

You can use the `GXPrintingAlert` function to create a printing alert box when you have not defined a resource for this purpose. This function takes as parameters all of the values that you would specify in a printing alert resource, draws the printing alert box, and displays it with any informative text you supply. You get the same results as you would by calling the `GXGetPrintingAlert` function with the ID of a printing alert that you defined in a resource.

Managing Paper Trays and Their Paper Types

QuickDraw GX allows the user to assign specific paper types to specific paper trays on a printer, as described in *Inside Macintosh: QuickDraw GX Printing*. You can use the tray-handling functions from within your message overrides to set or determine the paper type associated with each tray.

The paper trays on a printer are either configured or not. If one or more of the trays on the printer has a specific paper type assigned to it, then the trays are configured. If none of the trays has a specific paper type assigned to it, then the trays are unconfigured. You can call the `GXGetTrayMapping` function to discover if the printer with which your driver is communicating has trays that are configured.

If you need to cycle through the trays on a printer, you can call the `GXCountTrays` function to discover how many trays the printer has. For example, if you need to find which tray contains a certain paper type, you can call `GXCountTrays` and use the value it returns to limit your search.

If you want to determine if a certain paper type is in one of the paper trays, you can first call `GXGetTrayMapping`. If it tells you that the trays are not configured, than you don't need to look any farther. If it tells you that the trays are configured, you can call `GXCountTrays` to determine how many trays there are, and then loop through the trays.

You can determine the name of each tray by calling the `GXGetTrayName` function. You can set the paper type that is associated with a tray by calling the `GXSetTrayPaperType` function, and you can determine which paper type is associated with a tray by calling the `GXGetTrayPaperType` function.

Listing 5-1 shows how the LaserWriter printer driver fills in the paper-type names in the Printing menu by looping through the paper trays.

Listing 5-1 Looping through the paper trays

```
for (i = iEnvelopePopUp; i <= iLargePopUp; ++i)
   {
   GetDItem(dPtr, i, &theKind, &theHandle, &box);
   SetCtlMin((ControlHandle) theHandle, 1);
   SetCtlMax((ControlHandle) theHandle, CountMItems(theMenu));
   SetCtlValue((ControlHandle) theHandle, 1);

   /* if you don’t have it, hide it*/
   if    (
         ( (i == iEnvelopePopUp) && (!(options & kEnvelopeMask)) )
       ||( (i == iLargePopUp) && (!(options & k500SheetMask)) )
         )
      HideDItem(dPtr, i);
   else
      {
```

```
      anErr = GXGetTrayPaperType(i-iEnvelopePopUp+1, paper);
      if (anErr == noErr)
         {
         Str31    paperName, menuString;
         short    j;

         GXGetPaperTypeName(paper, paperName);

         for (j = CountMItems(theMenu); j > 1; --j)
            {
            GetItem(theMenu, j, menuString);
            if (IUCompString(menuString, paperName) == 0)
               {
               SetCtlValue((ControlHandle) theHandle, j);
               break;
               }
            }
         }

      /*
         If you don’t have the resource, it is not an error;
         it means that you have a fresh tray setup.
      */
      if (anErr == resNotFound)
         anErr = noErr;
      nrequire(anErr, FailedGetTrayPaperType);
      }
```

Storing and Accessing Data Associated With a Desktop Printer

Applications can store and access information in the data collections that are associated with a specific desktop printer. This allows you to preserve information with the desktop printer that your printing extension or printer driver can use when processing any document.

For example, you could discover that a printer has a color ribbon and store that fact with the desktop printer so that you could report it to the application. Or if the user has set a special option for the printer in the Printing menu, you might want to store that option with the desktop printer so that your driver could use it during printing.

QuickDraw GX provides two functions for interacting with the desktop printer data from within your message overrides. You use the `GXWriteDTPData` function to store data with the desktop printer, and you use the `GXFetchDTPData` function to access data that has been stored with the desktop printer.

Listing 5-2 shows the ImageWriter II printer driver's function for determining if the printer has a color ribbon. This function uses the `GXFetchDTPData` function and returns `true` if the printer does have a color ribbon.

Listing 5-2 Accessing the desktop printer data

```
Boolean PrinterHasColorRibbon(gxPrinter thePrinter)
{
   Boolean                 hasColor = true;
   Str32                   deviceName;
   OSErr                   anErr;
   ImageWriterConfigHandle configHandle;

   /*
      If not formatting to a particular device, then assume
      that there is a color ribbon, to allow for the widest
      range of color spaces.
   */
   GXGetPrinterName(thePrinter, deviceName);
   if (deviceName[0] != 0)
      {
      /*
         If formatting for a particular device, then assume that
         there is not a color ribbon because that is more common.
      */
      hasColor = false;

      anErr = GXFetchDTPData(deviceName, kImageWriterConfigType,
                  kImageWriterConfigID, &(Handle)configHandle);
      if (anErr == noErr)
         {
         hasColor = (**configHandle).hasColorRibbon;
         DisposHandle((Handle) configHandle);
         }
      }

   return(hasColor);

}
```

Providing Application Imaging Options

Application programs can provide the user with various print-quality and speed options. If you are writing a QuickDraw GX printing extension or printer driver, you may need to communicate to the application what options you support. These options come in two forms: printer view devices, which define the kinds of imaging that your driver can perform, and job format modes, which define the kinds of data interface that your printer driver supports.

You create a view device for each kind of imaging that your printer can support. You might provide a black-and-white view device and a color view device. You might provide a low-resolution view device and a high-resolution view device. You call the `GXAddPrinterViewDevice` function to add each of your supported view devices to the list of available view devices that the application can examine.

For example, the ImageWriter II printer driver sets a black-and-white 144 dpi view device and an 8-color 144 dpi view device in its override of the `GXDefaultPrinter` message. The code for this override is shown in Listing 3-4 on page 3-24 in the chapter “Printer Drivers.” QuickDraw GX view devices are described in *Inside Macintosh: QuickDraw GX Objects*.

Your printer driver also needs to communicate with the application about which job format modes the user can request. The job format mode enumeration is described on page 5-14. Each mode defines a kind of data with which your driver can operate, including graphics data, text-only data, and PostScript data. Job format modes are described in *Inside Macintosh: QuickDraw GX Printing*.

As a driver developer, you need to find out which job format modes the application provides to the user and then specify one of those modes as your preferred mode. You call the `GXGetAvailableJobFormatModes` function to determine which job format modes the application is providing. You then pick one of those modes as your preferred mode and call the `GXSetPreferredJobFormatMode` function to communicate your preference to the application. The preferred mode is the mode that gets set when the user selects direct-mode printing.

For example, in its override of the `GXJobDefaultFormatDialog` message, the ImageWriter II printer driver loops through the job format modes supported by the application to determine if the application supports text mode (for faster printing). If it does, the driver makes that the preferred mode. The code for this override is shown in Listing 3-7 on page 3-29 in the chapter “Printer Drivers.” Job format modes are described in *Inside Macintosh: QuickDraw GX Printing*.

Accessing Driver Data

Your driver sometimes needs to access data that belongs to the current print job. You can use the `GXGetJob` function to obtain a reference to the current job object.

You sometimes also need to access resource data that belongs to your driver. You can retrieve the resource file reference number for your driver by calling the `GXGetMessageHandlerResFile` function. You can also access your resource data by overriding the GXFetchTaggedData message, which is described on page 4-45 in the chapter “Printing Messages” in this book.

Interfacing With the Chooser

When the user selects your driver in the Chooser, the Chooser sends certain Chooser messages to you by calling the Device function, which is the interface for the package ('PACK') resource. You need to provide a version of this function to provide Chooser support for your driver. The package resource and its use are described in *Inside Macintosh: More Macintosh Toolbox*.

QuickDraw GX provides the GXHandleChooserMessage function, which takes care of most of the work of handling the Chooser package code for you. What you need to do is create a Device function, take appropriate action for messages that you need to handle, and pass the parameters on to the GXHandleChooserMessage function. Listing 5-3 shows the ImageWriter II printer driver implementation of the Device function.

Listing 5-3 The Device function for the ImageWriter II printer driver

```
pascal OSErr Device(short message, short caller,
                   StringPtr objName, StringPtr zoneName,
                   ListHandle theList, long p2)
{

   OSErr          anErr = noErr;
   extern Str31   gDriverName;
   StringPtr      pDriverName = &gDriverName;
   extern gxJob   gJob;
   gxJob          *pJob = &gJob;

   /* start up QuickDraw GX to begin with */
   if (message == initializeMsg)
      {
      FCBPBRec    pb;

      /* determine the driver name */
      pb.ioCompletion = nil;
      pb.ioNamePtr = pDriverName;
      pb.ioVRefNum = 0;
      pb.ioRefNum = CurResFile();
      pb.ioFCBIndx = 0;
      anErr = PBGetFCBInfo(&pb, false);

      *pJob = nil;
      if (anErr == noErr)
         {
         GXEnterGraphics();
         anErr = GXInitPrinting();
         if (anErr == noErr)
            anErr = GXNewJob(pJob);
         }
      }

   /* let the system handle the choosing */
   if (anErr == noErr)
      anErr = GXHandleChooserMessage(*pJob, pDriverName, message,
                        caller, objName, zoneName, theList, p2);

   /* shut down QuickDraw GX when done */
   if ( (message == terminateMsg) && (p2 == terminateMsg) )
      {
      if (*pJob != nil)
         GXDisposeJob(*pJob);
      GXExitPrinting();
      GXExitGraphics();
      }

   return(anErr);

}
```

The only Chooser messages that the ImageWriter II printer driver needs to do something special with are `initializeMsg` and `terminateMsg`. When the Chooser sends `initializeMsg`, the ImageWriter II driver initializes QuickDraw GX printing before passing the message along. And when the Chooser sends the `terminateMsg` message, the ImageWriter II driver terminates QuickDraw GX printing after passing the message along. The Chooser messages are described in *Inside Macintosh: Devices*.

Using the Message Cleanup Functions

When you override a message in your printing extension or printer driver, you often forward the message to other message handlers so that they can add their operations to the execution of the message.

Some QuickDraw GX message handlers allocate storage or modify state variables in response to certain printing messages. If something goes wrong in the processing of those messages, these handlers need to reverse their actions. The messages that require this reversing of actions, or cleanup, are the `GXOpenConnection`, `GXStartJob`, GXStartPage, and GXStartSendPage messages, which are described in the chapter “Printing Messages” in this book.

If you forward one of these messages from within your override code and then detect an error, you need to call the cleanup function. Each cleanup function sends a cleanup message to the handlers, starting with the handler immediately below your extension or driver in the chain. This ensures that each handler receives the message only once and in the appropriate order. The cleanup functions are: GXCleanupOpenConnection, GXCleanupStartJob, GXCleanupStartPage, and CleanupGXStartSendPage.

Listing 5-4 shows the ImageWriter II printer driver override of the GXOpenConnection message. This code uses the nrequire macro for exception handling and calls the GXCleanupOpenConnection function if an error occurs during the processing of the GXOpenConnection message. The nrequire macro is described in the chapter “Printer Drivers” in this book.

Listing 5-4 Calling the GXCleanupOpenConnection function

```
OSErr SD_OpenConnection(void)
{
   OSErr    anErr;


   /* first, open the connection the standard way */
   anErr = Forward_GXOpenConnection();
   nrequire(anErr, GXOpenConnection);

   /* then, bring the configuration file up to date */
   anErr = UpdateConfiguration();
   nrequire(anErr, UpdateConfiguration);

   return(noErr);

/* exception handling */
UpdateConfiguration:
   GXCleanupOpenConnection();

GXOpenConnection:

   return(anErr);

}
```

If an error occurs during the execution of the GXOpenConnection message, this override function calls the GXCleanupOpenConnection function.

Segmenting Your Driver Code

The smallest code segment that you can have with the messaging architecture consists of the code required to override a single message. However, in some cases you might want to have multiple code segments for a single message. For example, you might need a number of functions to override a message like `GXRenderPage`, which performs a lot of work.

QuickDraw GX provides a segment handler that you can use to divide your message override code into multiple segments. The segment handler allows you to construct a special 32-bit dispatch selector, which allows you to access functions in other code segments. The dispatch selector contains the segment's resource ID and the offset in that segment to the start of your function.

The `PRINTINGDISPATCH` macro constructs the dispatch selector and dispatch code for you, which allows you to call the function just as you would any other function. All that you need to do is define the macro. `PRINTINGDISPATCH` assumes that you start your segment with a long-aligned jump table, just like the jump tables that you use for your printing message overrides, as described in the section "The Jump Table" beginning on page 2-9 in the chapter "Printing Extensions." It also assumes that each entry in the table is 4 bytes long.

You define the dispatch macro with the segment ID and routine selector. For example, if your code is the third routine in a segment (it is at offset 12 in the segment) in a printer driver (`'pdvr'`) resource with an ID of 2, you define your macro as shown here:

```
OSErr MyRenderingRoutine (long param1, Ptr param2)
  = PRINTINGDISPATCH(2, 3);
```

All segment dispatches must return a value of type `OSErr`. If your function does not generate errors, you must still declare it to return an `OSErr` value and have the function return the constant `noErr`. You can then call your function as usual and also get free type-checking, as in this example:

```
   anErr = MyRenderingRoutine(p1, p2);   /* free type-checking */
```

You can call across segments by constructing your own 32-bit selector and calling the `GXPrintingDispatch` function directly. This method does not give you free type-checking and makes your code more complicated. However, the results are identical. For example, you could call the `MyRenderingRoutine` function using the following method:

```
  #define kMyRenderRoutineSelector 0x0002000C
   /* no type checking for the following call */
  anErr = GXPrintingDispatch(kMyRenderRoutineSelector, p1, p2);
```

The `GXPrintingDispatch` function is described on page 5-38.

Printing Functions Reference

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

The "Constants and Data Types" section shows the data structures and enumerated types that are used with the printing functions.

The "Functions" section describes the interface and functionality of each function that you can call only from within your overrides of the printing messages.

Constants and Data Types

This section describes the constants and data types that you use with the printing functions. Each of the types in this section is used as the type of a parameter to one or more of the printing functions.

Several data types that are used with the printing functions are described elsewhere. Notably, the `gxJob` and `gxPaperType` data types are described in *Inside Macintosh: QuickDraw GX Printing*.

Tray Index Type

The tray index type is used to designate a specific paper tray on a printer.

```
typedef long gxTrayIndex;
```

Tray Mapping Modes

The tray-mapping mode constants define the current tray-mapping mode of the printer. The `GXGetTrayMapping` function returns a value of a tray-mapping type, which you can use to determine if you need to loop through the trays to find information.

```
enum {
  gxDefaultTrayMapping     = (gxTrayMapping) 0,
  gxConfiguredTrayMapping  = (gxTrayMapping) 1
};

typedef long gxTrayMapping;
```

Constant descriptions

`gxDefaultTrayMapping`	None of the paper trays has a specific paper type assigned to it.
`gxConfiguredTrayMapping`	At least one of the paper trays has a specific paper type assigned to it.

Job Format Mode Table

The `GXGetAvailableJobFormatModes` function uses a job format mode table structure to communicate with the application about which job format modes are supported.

```
struct gxJobFormatModeTable {
   long              numModes;
   gxJobFormatMode   modes[1];
};

typedef struct gxJobFormatMode Table gxJobFormatModeTable,
*gxJobFormatModeTablePtr, **gxJobFormatModeTableHdl;
```

Field descriptions

`numModes`	The number of the entries in the `modes` array that follows.
`modes`	An array of mode constants. This array can be of any length. The constants are defined in the job format modes enumeration, which is defined in the next section.

Job Format Modes

The job format modes enumeration defines the constants used in the `modes` field of the job format mode table, which is described in the previous section. The constants define the current data mode of the printer.

```
enum {
   gxGraphicsJobFormatMode  = (gxJobFormatMode) 'grph',
   gxTextJobFormatMode      = (gxJobFormatMode) 'text',
   gxPostScriptJobFormatMode= (gxJobFormatMode) 'post'
};

typedef OSType gxJobFormatMode;
```

Constant descriptions

`gxGraphicsJobFormatMode`	Standard graphics job-format mode. In this mode, a printer driver supports QuickDraw GX graphics and typography.
`gxTextJobFormatMode`	Text job-format mode. In this mode, a printer driver supports native printer fonts and horizontal and vertical lines.
`gxPostscriptJobFormatMode`	PostScript job-format mode. In this mode, a printer driver supports the PostScript language.

The Panel Setup Structure

The panel setup structure, of data type `gxPanelSetupRecord`, is passed to the `GXSetupDialogPanel` function when the user displays a dialog box.

```
struct gxPanelSetupRecord {
  gxPrintingPanelKind   panelKind;
  short                 panelResId;
  short                 resourceRefNum;
  void                  *refCon;
};

typedef struct gxPanelSetupRecord gxPanelSetupRecord;
```

Field descriptions

`panelKind`	The kind of program that is using this panel. This value is one of the constants defined in the printing panel kinds enumeration, which is described in the next section.
`panelResId`	The resource ID of the panel (`'ppnl'`) resource for the dialog box panel.
`resourceRefNum`	The resource file reference number for the panel.
`refCon`	A reference constant for use by the creator of the panel.

Printing Panel Kinds

The printing panel kinds enumeration provides constants for use in the `panelKind` field of the panel setup structure, which is described in the previous section.

```
enum {
  gxApplicationPanel= (gxPrintingPanelKind) 0,
  gxExtensionPanel  = (gxPrintingPanelKind) 1,
  gxDriverPanel     = (gxPrintingPanelKind) 2
};

typedef long gxPrintingPanelKind;
```

Constant descriptions

`gxApplicationPanel`	A panel created for an application.
`gxExtensionPanel`	A panel created for a printing extension.
`gxDriverPanel`	A panel created for a printer driver.

Functions

This section describes the functions that you can call only from within your overrides of the printing messages. The functions are divided into the following categories:

- The status and alert display functions are used to display feedback to the user during the process of printing a document.
- The tray-handling functions are used to manage which paper is assigned to each paper tray on the printer.
- The desktop-printer data functions are used to store and access resource information in a desktop printer.
- The dialog box panel function is used to manage the addition of panels to a print dialog box.
- The application imaging options functions are used to communicate with the application regarding the imaging options provided by your printing extension or printer driver.
- The printing control functions are used to manage contextual information about your printing extension or printer driver.
- The message cleanup functions are used to perform cleanup tasks when certain printing messages fail.
- The segment control function allows you to segment your version of a printing message override.

Reporting Information to the User

You use the status and alert display functions to report information to the user.

The `GXReportStatus` function displays information that does not require user feedback, such as messages that monitor the current state of the document being printed. Typical status information text strings include “2 pages remaining to print” and “Starting job.”

The `GXAlertTheUser` function displays information to the user in a printing alert box. The user must repond to the printing alert box. Typical printing alert text strings include “Printer is out of paper” and “Printer requires attention: paper jam.”

The `GXGetPrintingAlert` function displays a printing alert box that is defined in a printing alert (`'plrt'`) resource.

The `GXPrintingAlert` function builds a printing alert box from the supplied parameters and displays it on the user’s screen.

GXReportStatus

You can use the `GXReportStatus` function to display information to the user in the desktop printer window.

```
OSErr GXReportStatus ( short statusId,
                       unsigned short statusIndex);
```

`statusId`	The ID of the status (`'stat'`) resource that contains the status strings.
`statusIndex`	The index of the status string that you want displayed.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

You call the `GXReportStatus` function to display informational feedback to the user during the printing of a document. You need to specify the ID of the status resource in which the feedback string is stored.

If the information that you want to display requires user attention, you need to call the `GXAlertTheUser` function instead.

RESULT CODES

`gxSegmentLoadFailedErr`	A required code segment could not be found, or there was not enough memory to load it.
`gxPrUserAbortErr`	The user has canceled printing.
`gxUnknownAlertVersionErr`	The printing alert version specified in the resource is not valid.

SEE ALSO

You can find an example of using the `GXReportStatus` function in Listing 3-15 on page 3-42 in the chapter “Printer Drivers.”

You can read about status resources in the section “The Status ('stat') Resource” beginning on page 6-19 in the chapter “Printing Resources.”

You can read about displaying status information in the section “Displaying Status Information and the Printing Alert Boxes” beginning on page 5-4. You can also read about this task in the section “Displaying Status Information and the Printing Alert Boxes” beginning on page 3-41 in the chapter “Printer Drivers.”

GXAlertTheUser

You can use the `GXAlertTheUser` function to display a printing alert box to the user.

```
OSErr GXAlertTheUser (gxStatusRecord *status);
```

`status`	A status structure of type `gxStatusRecord`.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The `GXAlertTheUser` function displays status information to the user in a printing alert box. The information to be displayed is described in the `status` parameter. The user must explicitly dismiss the printing alert box, and you must monitor the user's actions while the alert box is displayed.

RESULT CODES

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

SEE ALSO

You can find examples of using the `GXAlertTheUser` function in Listing 3-15 on page 3-42 and in Listing 3-16 on page 3-44 in the chapter "Printer Drivers."

A description of how to use the `GXAlertTheUser` function, including an example from the ImageWriter II driver, is found in the section "Displaying Status Information and the Printing Alert Boxes" beginning on page 3-41 in the chapter "Printer Drivers."

The `gxStatusRecord` structure is described in the section "The Status Structure" beginning on page 4-39 in the chapter "Printing Messages."

GXGetPrintingAlert

You can use the `GXGetPrintingAlert` function to display a printing alert box defined by a printing alert resource.

```
OSErr GXGetPrintingAlert (short alertResId,
                    ModalFilterProcPtr filterProc, short *itemHit);
```

`alertResId`	The ID of the printing alert (`'plrt'`) resource that defines the contents of the alert box.

`filterProc`	A pointer to the function to use for filtering events that occur while the alert box is displayed.
`itemHit`	A pointer to the ID of the item that the user selected to dismiss the alert box.

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

DESCRIPTION

The `GXGetPrintingAlert` function displays a printing alert box. The contents of the alert box are defined in the printing alert resource that has the ID specified in the `alertResId` parameter.

User-initiated events that occur while the alert box is displayed are filtered through the function that is specified by the `filterProc` parameter. When the user dismisses the printing alert box by selecting an item, this function returns in the `itemHit` parameter the ID of the item that was selected.

RESULT CODES

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

SEE ALSO

The printing alert resource is described in the section “The Printing Alert ('plrt') Resource” beginning on page 6-21 in the chapter “Printing Resources”.

You can use the `GXPrintingAlert` function, as described in the next section, to display a printing alert box whose contents are defined as parameters to the function.

GXPrintingAlert

You can use the `GXPrintingAlert` function to create and display a printing alert box. This function creates the alert box from its parameters. The `GXGetPrintingAlert` function, which is described in the previous section, displays a printing alert box that is created from a resource definition.

```
OSErr GXPrintingAlert (short iconId, short txtSize,
                  short defaultTitleNum, short cancelTitleNum,
                  short textLength, Ptr pAlertMsg,
                  StringPtr actionTitle, StringPtr title2,
                  StringPtr title3, StringPtr msgFont,
                  ModalFilterProcPtr filterProc, short *itemHit,
                  StringPtr alertTitle);
```

`iconId`	The type of icon to display in the printing alert box. Use one of the following constants for the value of this parameter: `noIcon`, `stopIcon`, `cautionIcon`, or `noteIcon`.
`txtSize`	The font size of the text to display in the printing alert box. You can use the value `defaultSystemSize` to indicate that the default text size for the system is to be used.
`defaultTitleNum`	The type of text string to display on the default button. Use one of the following constants as the value of this parameter: `noDefaultTitle`, `defaultAction`, `defaultTitle2`, or `defaultTitle3`.
`cancelTitleNum`	The type of text string to display on the Cancel button. Use one of the following constants as the value of this parameter: `noCancelTitle`, `cancelAction`, `cancelTitle2`, or `cancelTitle3`.
`textLength`	The length of the text string that is to be displayed in the printing alert box.
`pAlertMsg`	A pointer to the message text to display in the printing alert box.
`actionTitle`	A pointer to the string to display on the action button.
`title2`	A pointer to the string to display on button 2.
`title3`	A pointer to the string to display on button 3.
`msgFont`	A pointer to a string that names the font to use for displaying the text string in the printing alert box.
`filterProc`	A pointer to the function to use for filtering events that occur while the alert is displayed.
`itemHit`	On return, the ID of the item that was selected by the user to dismiss the alert box.
`alertTitle`	The string that is displayed in the title bar of the printing alert box.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The `GXPrintingAlert` function displays a printing alert box, building it from the parameters that you pass instead of from a printing alert (`'plrt'`) resource.

User-initiated events that occur while the alert box is displayed are filtered through the function that is specified by the `filterProc` parameter. When the user dismisses the alert box by selecting an item, this function returns in the `itemHit` parameter the ID of the item that was selected.

Each of the parameters to this function matches a field in the printing alert resource.

RESULT CODES

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

SEE ALSO

The printing alert resource is described in the section “The Printing Alert ('plrt') Resource” beginning on page 6-21 in the chapter “Printing Resources.”

You can use the `GXGetPrintingAlert` function, which is described on page 5-18, to display a printing alert box that is created from a resource definition.

Managing Paper Trays

You use the tray-handling functions to manage the use of the paper trays on the printer with which your printing extension or printer driver is communicating. Each tray on the printer is named, and each can have a specific paper type assigned to it.

You use the `GXCountTrays` function when you need to loop through the trays on the printer and determine which kind of paper is assigned to each. The `GXCountTrays` function tells you how many trays you need to search.

The `GXGetTrayName` function returns the name of a specific tray on the printer. The `GXGetTrayMapping` function tells you whether or not any of the trays on the printer has been assigned a specific paper type. If not, then you don’t need to loop through them to find an assigned type.

The `GXSetTrayPaperType` function allows you to associate a specific paper type with a specific tray on the printer, and the `GXGetTrayPaperType` function allows you to access the paper-type information that is associated with a specific tray.

GXCountTrays

You can use the `GXCountTrays` function to determine the number of paper trays that are available for your printing extension or printer driver to use.

```
OSErr GXCountTrays (gxTrayIndex *numTrays);
```

`numTrays`	A pointer to a value that, on return, contains the number of paper trays that are currently available on the printer.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The `GXCountTrays` function calls the `GXFetchTaggedData` function to retrieve the number of paper trays that are currently available on the printer. You call the `GXCountTrays` function to determine how many trays there are when you need to loop through the paper trays to look for information.

RESULT CODES

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

SEE ALSO

The `GXFetchTaggedData` function is described in *Inside Macintosh: QuickDraw GX Environment and Utilities.*

You can read about managing paper trays in the section "Managing Paper Trays and Their Paper Types" beginning on page 5-5.

GXGetTrayName

You can use the `GXGetTrayName` function to retrieve the name of a specified paper tray on the printer.

```
OSErr GXGetTrayName (gxTrayIndex whichTray, Str31 trayName);
```

`whichTray`	The number of the tray in which you are interested.
`trayName`	On return, the name of the specified tray. This parameter is a string that you have already allocated.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The `GXGetTrayName` function fills in the `trayName` string with the name of the specified tray on the printer. This information is retrieved by calling the `GXFetchTaggedData` function.

RESULT CODES

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

SEE ALSO

The `GXFetchTaggedData` function is described in *Inside Macintosh: QuickDraw GX Environment and Utilities.*

You can read about managing paper trays in the section "Managing Paper Trays and Their Paper Types" beginning on page 5-5.

GXGetTrayMapping

You can use the `GXGetTrayMapping` function to determine the current tray-mapping mode of the printer.

```
OSERR GXGetTrayMapping (gxTrayMapping *trayMapping);
```

`trayMapping`
A pointer to a value that, on return, is the current tray-mapping mode of the printer.

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

DESCRIPTION

The `GXGetTrayMapping` function returns the current tray-mapping mode of the printer in the `trayMapping` parameter. If any of the paper trays has an assigned paper type, the value of `trayMapping` is set to `gxConfiguredTrayMapping`; otherwise, the value of `trayMapping` is set to `gxDefaultTrayMapping`.

You can use this function to determine if you need to loop through the paper trays to find a certain paper type. If the returned value of `trayMapping` is `gxConfiguredTrayMapping`, you know that at least one of the trays contains a specific paper type and that you do need to loop through the trays. If the returned value of `trayMapping` is `gxDefaultTrayMapping`, none of the trays has a specific paper type assigned to it, so you don't need to loop through them.

RESULT CODES

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

SEE ALSO

The `gxTrayMapping` enumeration is described on page 5-13.

You can read about managing paper trays in the section "Managing Paper Trays and Their Paper Types" beginning on page 5-5.

GXSetTrayPaperType

You can use the `GXSetTrayPaperType` function to change the paper-type information for a specific paper tray on the printer.

```
OSErr GXSetTrayPaperType (gxTrayIndex whichTray,
                                gxPaperType paper);
```

`whichTray`	The number of the tray for which you are changing the paper-type information.
`paper`	A reference to a paper-type object that contains information about the paper that is in the specified tray. If you specify `nil` as the value of this parameter, the tray becomes unconfigured.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The `GXSetTrayPaperType` function modifies the paper-type information that is associated with the specified tray on the printer. The information that you supply in the `paper` parameter is stored with the desktop printer and is used to facilitate the handling of mismatched paper types on the printer.

RESULT CODES

`gxSegmentLoadFailedErr`	A required code segment could not be found, or there was not enough memory to load it.
`gxPrUserAbortErr`	The user has canceled printing.
`gxPaperTypeNotFound`	The specified paper type could not be found.
`gxNoSuchPTGroup`	The specified paper type could not be found.

SEE ALSO

You can read about managing paper trays in the section "Managing Paper Trays and Their Paper Types" beginning on page 5-5.

GXGetTrayPaperType

You can use the `GXGetTrayPaperType` function to retrieve the paper-type information for a specific paper tray on the printer.

```
OSErr GXGetTrayPaperType (gxTrayIndex whichTray,
                          gxPaperType paper);
```

`whichTray`	The number of the tray in which you are interested.
`paper`	A reference to a paper-type object that you have already allocated. On return, this object contains information about the paper that is in the specified tray.
function result	An error code. The values `noErr` indicates that the operation was successful. The value `resNotFound` indicates that the tray is not configured and that its paper type is displayed to the user as "Unknown."

DESCRIPTION

The `GXGetTrayPaperType` function fills in the fields of a paper-type object with information from the desktop printer.

RESULT CODES

`gxSegmentLoadFailedErr`	A required code segment could not be found, or there was not enough memory to load it.
`gxPrUserAbortErr`	The user has canceled printing.
`gxPaperTypeNotFound`	The specified paper type could not be found.
`gxNoSuchPTGroup`	The specified paper type could not be found.

SEE ALSO

You can read about managing paper trays in the section "Managing Paper Trays and Their Paper Types" beginning on page 5-5.

Storing and Accessing Desktop Printer Data

You use the desktop printer data functions to store and access information in the data collections associated with a specific desktop printer. You store information with the desktop printer when you want that information to be accessible across multiple print jobs.

Desktop printers are described in *Inside Macintosh: QuickDraw GX Printing*.

GXWriteDTPData

You can use the `GXWriteDTPData` function to store data in a desktop printer.

```
OSErr GXWriteDTPData (Str31 dtpName, OSType theType,
                      short theID, Handle theData);
```

`dtpName`	The name of the desktop printer in which the data is to be stored.
`theType`	The type of the data that you are storing.
`theID`	The ID of the data that you are storing.
`theData`	A handle to the data.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The `GXWriteDTPData` function stores data in the desktop printer.

You supply the name of the printer in the `dtpName` parameter, the type in the `theType` parameter, and the ID in the `theID` paramter. `GXWriteDTPData` copies the data from the handle specified by the `theData` parameter.

You must call the `DisposeHandle` function on the handle after calling this function.

RESULT CODES

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

SEE ALSO

You can find an example of using the `GXWriteDTPData` function in Listing 3-9 on page 3-31 in the chapter “Printer Drivers.”

The `DisposeHandle` function is described in *Inside Macintosh: Memory*.

GXFetchDTPData

You can use the `GXFetchDTPData` function to access data from the desktop printer.

```
OSErr GXFetchDTPData (Str31 dtpName, OSType theType,
                      short theID, Handle *theData);
```

`dtpName`	The name of the desktop printer from which to retrieve the data.
`theType`	The type in which you are interested.
`theID`	The ID in which you are interested.
`theData`	A pointer to a handle that, on return, references the data.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The `GXFetchDTPData` function accesses data from the desktop printer. You can call this function if you need to examine the data. `GXFetchDTPData` accesses the data and returns a handle to it in the `theData` parameter.

The handle that you receive back from this function has been detached, which means that you must call the `DisposeHandle` function on it when you have finished with it.

RESULT CODES

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

SEE ALSO

You can find an example of using the `GXFetchDTPData` function in Listing 3-9 on page 3-31 in the chapter “Printer Drivers.”

The `DisposeHandle` function is described in *Inside Macintosh: Memory.*

Adding a Panel to a Print Dialog Box

You use the dialog box panel function, `GXSetupDialogPanel`, to add a panel to a print dialog box.

GXSetupDialogPanel

You can use the `GXSetupDialogPanel` function to add a panel to a print dialog box.

```
OSErr GXSetupDialogPanel (gxPanelSetupRecord *panelRec);
```

`panelRec`	A pointer to a panel setup structure.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The `GXSetupDialogPanel` function adds a panel, as defined by the information in the panel setup structure, to a print dialog box. You call this function from within your override of the `GXJobPrintDialog`, `GXFormatDialog`, and `GXJobDefaultFormatDialog` messages, before forwarding the message.

RESULT CODES

`gxSegmentLoadFailedErr`	A required code segment could not be found, or there was not enough memory to load it.
`gxPrUserAbortErr`	The user has canceled printing.
`gxCantAddPanelsNowErr`	Panels can only be added to a dialog box when the current driver is switched. This error is generated if a panel addition request is made at any other time.
`gxBadxdtlKeyErr`	An unknown key value was specified for an item in an extended dialog control resource.
`gxXdtlItemOutOfRangeErr`	An item referenced by the panel does not belong to the panel.
`gxNoActionButtonErr`	The action button for the panel is `nil`.
`gxTitlesTooLongErr`	The length of the button titles exceeds the maximum width allowed for a printing alert.

SEE ALSO

You can find an example of using the `GXSetupDialogPanel` function in Listing 2-10 on page 2-18 in the chapter “Printing Extensions.”

The `gxPanelSetupRecord` structure is described in the section “The Panel Setup Structure” on page 5-15.

The `GXJobPrintDialog`, `GXFormatDialog`, and `GXJobDefaultFormatDialog` messages are described in the chapter “Printing Messages” in this book.

Working With Application Imaging Options

You use the application imaging-option functions in your printer driver to communicate with the application regarding which imaging options are available for printing documents. These options include which kinds of view devices your driver supports and which job format mode your driver prefers.

GXAddPrinterViewDevice

You can use the `GXAddPrinterViewDevice` function to add a view device to the list of view devices that can be used with a printer.

```
OSErr GXAddPrinterViewDevice (gxPrinter printer,
                              gxViewDevice viewDev);
```

`printer`	A reference to a printer object.
`viewDev`	The view device to associate with the printer.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The `GXAddPrinterViewDevice` function associates a view device with a specified printer. You call this function from your printer driver to provide information to the application about the kinds of imaging that your printer driver supports for the printer. For example, you can make a view device that is 32 bits deep and associate that with the printer to communicate to the application that your printer driver can manage imaging that is 32 bits deep.

This function increments the owner count of the view device, which means that you still need to call the `GXDisposeViewDevice` function after calling this function.

RESULT CODES

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

SEE ALSO

You can find an example of using the `GXAddPrinterViewDevice` function in Listing 3-5 on page 3-25 in the chapter “Printer Drivers.”

The `GXDisposeViewDevice` function is described in I*nside Macintosh: QuickDraw GX Objects*.

GXGetAvailableJobFormatModes

You can use the `GXGetAvailableJobFormatModes` function to determine which job format modes the application supports.

```
OSErr GXGetAvailableJobFormatModes (
                              gxJobFormatModeTableHdl *modeTbl);
```

`modeTbl`	A pointer to a `gxJobFormatModeTable` structure.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

You call the `GXGetAvailableJobFormatModes` function to find out which of the job format modes the application supports. The application establishes these modes by calling the `GXSetPreferredJobFormatMode` function.

Your printer driver calls this function and you pick one of the modes that the application supports as the preferred mode for your printer driver. You then call the `GXSetPreferredJobFormatMode` function to establish that mode as the preference for the application.

RESULT CODES

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

SEE ALSO

You can find an example of using the `GXGetAvailableJobFormatModes` function in Listing 3-7 on page 3-29 in the chapter "Printer Drivers."

The `gxJobFormatModeTable` structure is described on page 5-14.

The `GXSetPreferredJobFormatMode` function is described in the next section.

GXSetPreferredJobFormatMode

You can use the `GXSetPreferredJobFormatMode` function to define, for the application , which job format mode is the preferred mode for your printer driver.

```
OSErr GXSetPreferredJobFormatMode (gxJobFormatMode mode,
                                   Boolean directOnly);
```

`mode`	The job format mode that your driver prefers.
`directOnly`	A Boolean value that is `true` if your driver only supports text mode and `false` if not.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

The `GXSetPreferredJobFormatMode` function establishes which of the job format modes is the preferred mode of a printer driver. If a printer driver supports PostScript, it generally sets `gxPostscriptJobFormatMode` as its preferred mode. If a driver only supports the use of built-in text fonts, it sets `gxTextJobFormatMode` as its preferred mode. Otherwise, a printer driver generally sets `gxGraphicsJobFormatMode` as its preferred mode.

You set the `directOnly` value to `true` if your printer driver does not support graphics mode. For instance, this is the case for a daisy-wheel printer. In this case, you communicate to the application that it cannot send graphics to be printed by your driver by setting the `directOnly` value to `true`.

RESULT CODES

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

SEE ALSO

You can find an example of using the `GXSetPreferredJobFormatMode` function in Listing 3-7 on page 3-29 in the chapter “Printer Drivers.”

The job format mode constants are described on page 5-14.

Printing Control Functions

The printing control functions are a collection of miscellaneous functions that you can use within the implementations of your printing message overrides. They include

- the `GXGetJob` function, which you can call to obtain the job object for the current printing job that your printing extension or printer driver is handling
- the `GXGetMessageHandlerResFile` function, which you can call to obtain the resource file reference number for your printing extension or printer driver
- the `GXSpoolingAborted` function, which you can call to determine if the spooling of a document has been aborted
- the `GXJobIdle` function, which you can call to release idle time to other processes

- the `GXHandleChooserMessage` function, which you can call to handle the Chooser messages that are sent to your printer driver

GXGetJob

You can use the `GXGetJob` function to retrieve the job object that is currently associated with your printing extension or printer driver.

```
gxJob GXGetJob (void);
```

function result The job object for the current print job.

DESCRIPTION

The `GXGetJob` function returns the current print job as its function result. You can call this function if you need to access the information that is associated with the job object, including its collections.

RESULT CODES

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

GXGetMessageHandlerResFile

You can use the `GXGetMessageHandlerResFile` function to retrieve the resource file reference number of your printing extension or printer driver.

```
short GXGetMessageHandlerResFile (void);
```

function result The resource file reference number of your printing extension or printer driver.

DESCRIPTION

The `GXGetMessageHandlerResFile` function returns the resource file reference number for your extension or driver. You can use this function if you need to access data from your resources.

RESULT CODES

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

GXJobIdle

You can use the `GXJobIdle` function to release time to other processes while your printing extension or printer driver is performing a computationally intensive task.

```
OSErr GXJobIdle (void);
```

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

DESCRIPTION

The `GXJobIdle` function tells QuickDraw GX to release time to other processes that are currently active. If your driver 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

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

GXSpoolingAborted

You can use the `GXSpoolingAborted` function to determine if a user has cancelled spooling of a document.

```
Boolean GXSpoolingAborted (void);
```

function result A Boolean value. A value of `true` indicates that the current printing job has been aborted. A value of `false` indicates that it has not been aborted.

DESCRIPTION

The `GXSpoolingAborted` function returns a Boolean value as its function result. This value tells you if spooling of the current document has been aborted.

RESULT CODES

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

GXHandleChooserMessage

You can use the `GXHandleChooserMessage` function to respond when a user selects a Chooser item for your printer driver.

```
OSErr GXHandleChooserMessage (gxJob *aJob, Str31 driverName,
                              short message, short caller,
                              StringPtr objName, StringPtr zoneName,
                              ListHandle theList, long p2);
```

`aJob`	A reference to a job object.
`driverName`	The name of the printer driver; for example, “LaserWriter.”
`message`	The operation that is to be performed.
`caller`	The caller. A value of 1 indicates the Chooser. Values in the range 0 to 127 are reserved; values outside of this range can be used by applications.
`objName`	A pointer to other information. The value of this parameter depends on the value of the `message` parameter.
`zoneName`	The name of the AppleTalk zone containing the devices in the device list.
`theList`	A handle to the List Manager list that contains the device choices that are displayed in the device list box.
`p2`	Other information. The meaning of this parameter depends on the value of the `message` parameter.
function result	An error code. The value `noErr` indicates that the operation was successful.

DESCRIPTION

When the user interacts with your driver in the Chooser, the Chooser sends messages to your driver. What the Chooser sends depends on information in the package (`'PACK'`) resource that you have defined for your driver. The `GXHandleChooserMessage` function makes it possible for your driver to work with the Chooser without you having to write the package-handling code.

When the Chooser sends a message to your driver, you can call this function to allow QuickDraw GX to handle the Chooser actions for you. You can perform your own actions, depending on which message has been sent, and then pass the information that the Chooser sent to you along to this function, which takes care of the rest for you.

The messages that the Chooser can send to your driver are shown in Table 5-1.

Table 5-1 Messages that the Chooser sends to drivers

Message	Value	Explanation
`init`	11	The Chooser sends this message to your package when the user selects the icon representing your package in the icon list.
`newSel`	12	If your device package allows multiple selections, the Chooser sends this message to your packages when the user changes or adds a selection.
`fillList`	13	The Chooser sends this message when the user selects a device icon.
`getSel`	14	The Chooser sends this message to determine which entries should be selected. The Chooser does not send this message for serial printers.
`select`	15	If your device package does not allow multiple selections, the Chooser sends this message to your packages when the user selects a device from the device list.
`deselect`	16	If your device package does not allow multiple selections, the Chooser sends this message to your packages when the user deselects a device in the device list.
`terminate`	17	The Chooser sends this message when the user selects a different device icon, closes the Chooser window, or changes zones.
`button`	18	The Chooser sends this message when the user clicks one of the buttons in the Chooser window.

RESULT CODES

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

SEE ALSO

An example of using the `GXHandleChooserMessage` function is shown in Listing 5-3 on page 5-9.

Inside Macintosh: Devices describes the Chooser messages and package resources in detail.

Handling Error Conditions in a Message Override

You use the message cleanup functions when you encounter an error during the processing of certain printing messages that change the state of your driver. When you call a cleanup function, QuickDraw GX sends a corresponding cleanup message to all of the handlers in the message chain, allowing them to revert to their previous state.

GXCleanupOpenConnection

You can use the `GXCleanupOpenConnection` function to notify QuickDraw GX that it needs to clean up any actions that it took after receiving a `GXOpenConnection` message from your printing extension or printer driver.

```
void GXCleanupOpenConnection (void);
```

DESCRIPTION

You need to call the `GXCleanupOpenConnection` function if you encounter an error in your override of the `GXOpenConnection` message and you have already forwarded the message.

When you call `GXCleanupOpenConnection`, QuickDraw GX sends the `GXCleanupOpenConnection` message, which allows all of the message handlers to reverse any actions that they took in their overrides of the `GXOpenConnection` message, such as allocating storage or modifying state information.

SEE ALSO

The `GXOpenConnection` and `GXCleanupOpenConnection` messages are described in the chapter "Printing Messages."

An example of using the `GXCleanupOpenConnection` function is shown in Listing 5-4 on page 5-11.

GXCleanupStartJob

You can use the `GXCleanupStartJob` function to notify QuickDraw GX that it needs to clean up any actions that it took after receiving a `GXStartJob` message from your printing extension or printer driver.

```
void GXCleanupStartJob (void);
```

DESCRIPTION

You need to call the `GXCleanupStartJob` function if you encounter an error in your override of the `GXStartJob` message and you have already forwarded the message.

When you call `GXCleanupStartJob`, QuickDraw GX sends the `GXCleanupStartJob` message, which allows all of the message handlers to reverse any actions that they took in their overrides of the `GXStartJob` message, such as allocating storage or modifying state information.

SEE ALSO

The `GXStartJob` and `GXCleanupStartJob` messages are described in the chapter "Printing Messages."

GXCleanupStartPage

You can use the `GXCleanupStartPage` function to notify QuickDraw GX that it needs to clean up any actions that it took after receiving a `GXStartPage` message from your printing extension or printer driver.

```
void GXCleanupStartPage (void);
```

DESCRIPTION

You need to call the `GXCleanupStartPage` function if you encounter an error in your override of the `GXStartPage` message and you have already forwarded the message.

When you call `GXCleanupStartPage`, QuickDraw GX sends the `GXCleanupStartPage` message, which allows all of the message handlers to reverse any actions that they took in their overrides of the `GXStartPage` message, such as allocating storage or modifying state information.

SEE ALSO

The `GXStartPage` and `GXCleanupStartPage` messages are described in the chapter "Printing Messages."

GXCleanupStartSendPage

You can use the `GXCleanupStartSendPage` function to notify QuickDraw GX that it needs to clean up any actions that it took after receiving a `GXStartSendPage` message from your printing extension or printer driver.

```
void GXCleanupStartSendPage (void);
```

DESCRIPTION

You need to call the `GXCleanupStartSendPage` function if you encounter an error in your override of the `GXStartSendPage` message and you have already forwarded the message.

When you call `GXCleanupStartSendPage`, QuickDraw GX sends the `GXCleanupStartSendPage` message, which allows all of the message handlers to reverse any actions that they took in their overrides of the `GXStartSendPage` message, such as allocating storage or modifying state information.

SEE ALSO

The `GXStartSendPage` and `GXCleanupStartSendPage` messages are described in the chapter "Printing Messages."

Segmenting Message Override Code

The `GXPrintingDispatch` function provides you with a means of segmenting your driver code into smaller boundaries than are permitted by the messaging architecture.

GXPrintingDispatch

You can use the `GXPrintingDispatch` function to make calls across segment boundaries.

```
OSErr GXPrintingDispatch (long selector, ...);
```

`selector`	The selector for the printing message that you want to dispatch.
`...`	The parameters that the message expects.

DESCRIPTION

The `GXPrintingDispatch` function calls a function that is located in a different code segment. You need to do this if the code to implement a single message override is too large to fit into one segment. This function uses the same information as does the `PRINTINGDISPATCH` macro, which provides a simpler technique for calling across segment boundaries. To use the `GXPrintingDispatch` function, you construct your own selector, as described on page 5-12, and use it as a parameter to the function.

SEE ALSO

An example of using the `GXPrintingDispatch` function is shown on page 5-12.

The `PRINTINGDISPATCH` macro method of calling across segments is described in the section "Segmenting Your Driver Code" beginning on page 5-12.

Summary of Printing Functions

Constants and Data Types

Tray Index Type

```
typedef long gxTrayIndex;        /* specifies a paper tray */
```

Tray Mapping Modes

```
enum {
   gxDefaultTrayMapping    = (gxTrayMapping) 0,    /* no tray has a defined
                                                       paper type assigned */
   gxConfiguredTrayMapping = (gxTrayMapping) 1     /* at least 1 tray has a
                                                       paper type assigned */
   };

typedef long gxTrayMapping;
```

Job Format Modes

```
enum {
   gxGraphicsJobFormatMode    = (gxJobFormatMode) 'grph',   /* graphics
                                                                and text */
   gxTextJobFormatMode        = (gxJobFormatMode) 'text',   /* text */
   gxPostScriptJobFormatMode  = (gxJobFormatMode) 'post'    /* PostScript */
};

typedef OSType gxJobFormatMode;
```

Job Format Mode Table

```
struct gxJobFormatModeTable {
   long              numModes;         /* number of entries in modes field */
   gxJobFormatMode   modes[1];         /* array of mode constants */
};

typedef struct gxJobFormatModeTable gxJobFormatModeTable,
*gxJobFormatModeTablePtr, **gxJobFormatModeTableHdl;
```

The Panel Setup Structure

```
struct gxPanelSetupRecord {
  gxPrintingPanelKind   panelKind;        /* kind of program using panel */
  short                 panelResId;       /* resource ID of panel */
  short                 resourceRefNum;   /* resource file refnum of panel */
  void                  *refCon;       /* pointer to panel setup
                                           structure used to build panel */
};

typedef struct gxPanelSetupRecord gxPanelSetupRecord;
```

Printing Panel Kinds

```
enum {
  gxApplicationPanel= (gxPrintingPanelKind) 0, /* an application panel */
  gxExtensionPanel  = (gxPrintingPanelKind) 1, /* printing extension panel */
  gxDriverPanel     = (gxPrintingPanelKind) 2  /* printer driver panel */
};

typedef long gxPrintingPanelKind;
```

Functions

Reporting Information to the User

```
OSErr GXReportStatus        (short statusID, unsigned short statusIndex);
OSErr GXAlertTheUser        (gxStatusRecord *status);
OSErr GXGetPrintingAlert    (short alertResId,
                             ModalFilterProcPtr filterProc, short *itemHit);
OSErr GXPrintingAlert       (short iconId, short txtSize,
                             short defaultTitleNum, short cancelTitleNum,
                             short textLength, Ptr pAlertMsg,
                             StringPtr actionTitle, StringPtr title2,
                             StringPtr title3, StringPtr msgFont,
                             ModalFilterProcPtr filterProc, short *itemHit,
                             StringPtr alertTitle);
```

Managing Paper Trays

```
OSErr GXCountTrays          (gxTrayIndex *numTrays);
OSErr GXGetTrayName         (gxTrayIndex whichTray, Str31 trayName);
OSErr GXGetTrayMapping      (gxTrayMapping *trayMapping);
OSErr GXSetTrayPaperType    (gxTrayIndex whichTray, gxPaperType paper);
OSErr GXGetTrayPaperType    (gxTrayIndex whichTray, gxPaperType paper);
```

Storing and Accessing Desktop Printer Data

```
OSErr GXWriteDTPData        (Str31 dtpName, OSType theType,
                             short theID, Handle theData);
OSErr GXFetchDTPData        (Str31 dtpName, OSType theType,
                             short theID, Handle *theData);
```

Adding a Panel to a Print Dialog Box

```
OSErr GXSetupDialogPanel    (gxPanelSetupRecord *panelRec);
```

Working With Application Imaging Options

```
OSErr GXAddPrinterViewDevice(gxPrinter printer, gxViewDevice viewDev);
OSErr GXGetAvailableJobFormatModes
                            (gxJobFormatModeTableHdl *modeTbl);
OSErr GXSetPreferredJobFormatMode
                            (gxJobFormatMode, Boolean directOnly);
```

Printing Control Functions

```
gxJob GXGetJob              (void);
short GXGetMessageHandlerResFile
                            (void);
Boolean GXSpoolingAborted   (void);
OSErr GXJobIdle             (void);
OSErr GXHandleChooserMessage
                            (gxJob *aJob, Str31 driverName,
                             short message, short caller,
                             StringPtr objName, StringPtr zoneName,
                             ListHandle theList, long p2);
```

Handling Error Conditions in a Message

```
void GXCleanupOpenConnection(void);
void GXCleanupStartJob      (void);
void GXCleanupStartPage     (void);
void GXCleanupStartSendPage (void);
```

Segmenting Message Override Code

```
OSErr GXPrintingDispatch    (long selector, ...);
```

C H A P T E R 6

Printing Resources

Contents

Printing Resources

This chapter describes the contents and related data structures for each of the resources you need to define for printing extensions and printer drivers. This chapter consists primarily of reference information. For specific information on using each resource in the context of developing an extension or driver, see the “Printing Extensions” and “Printer Drivers” chapters in this book.

You need to read this chapter if you are developing a printing extension or a printer driver for use with QuickDraw GX. This chapter constitutes the complete reference guide to the resources you can use to develop these programs.

Before reading this chapter, you need to have a basic understanding of Macintosh resources and the Macintosh Resource Manager, which are described in *Inside Macintosh: More Macintosh Toolbox*.

This chapter begins with a brief overview of the resources that QuickDraw GX uses for printing and then describes

- the numbers that you can use to identify resources
- the attributes that you specify in every resource
- information about file type and creator type for a resource
- each resource that you use for your extensions and drivers, including a description of each field in each of the resources

About the Printing Resources

Each printing extension and printer driver that you develop consists of a number of resources that you must create and package together. Some of the resources are standard Macintosh resources for the icons and panels that are part of the user interface to your extension or driver. Other resources are used specifically for QuickDraw GX printing.

Some of the resources described here are used for both printing extensions and printer drivers, some are used only for printing extensions, and some are used only for printer drivers. You define some resources for a printing extension or printer driver that you develop for any imaging system, and you define others only if your extension or driver is designed for a specific imaging system. The available imaging systems are raster, PostScript, and vector.

These resources contain information such as

- code to override message
- user interface data, such as dialog boxes and icons
- data that defines how to manage color handling on a specific device
- data used to establish appropriate communications with a device
- strings displayed to a user during printing
- paper types and formats used for printing

Resource ID Numbering

You assign a unique identifier, or resource ID, to each resource that you use in a Macintosh program. The printing resources that you define for printing extensions need to be in a certain range: from 0x9600 (-27136) to 0x97FF (-26625).

The printing resources that you define for printer drivers need to be in a certain range: from 0x9400 (-27648) to 0x95FF (-27137). For most of the printing-specific resources in your drivers, you define the ID as a value added to a constant named `gxPrintingDriverBaseID` (-28672), as in this example:

```
resource gxOverrideType (gxPrintingDriverBaseID+1, sysHeap,
                                                   purgeable)
{
   {
   gxRasterPackageBitmap, segmentID, firstOffset+40,
   gxRasterLineFeed,    segmentID, firstOffset+44
   };
};
```

QuickDraw GX defines constants for the resource ID of many of the resources that you define for your extensions and drivers. These constants are described in the reference description for each resource.

Resource Attributes

All of the resources that you define for your printing extensions and printer drivers need to be loaded into the system heap and need to be purgeable. System resources are stored in the system heap as opposed to the application heap, where application resources are stored. Purgeable resources can be purged by the Memory Manager when space is required, as described in *Inside Macintosh: Memory*.

You need to specify these attributes in the first line of every resource that you define for your extensions and drivers, as in this example:

```
resource gxOverrideType (gxPrintingDriverBaseID+1, sysHeap,
                                                   purgeable)
{
};

resource statusType (kDriverStatus, sysHeap, purgeable)
{
};
```

S **WARNING**
You need to store your resources in the system heap to allow the memory that they use to be shared when multiple copies of your driver are active at the same time. You must, however, take care to not release a resource that is in use by another copy of your driver. s

Extension and Driver Resource Files

Each printing extension and printer driver is implemented in a file of resources. Code segments of a printing extension must have the file type `'pext'`, and code segments of a printer driver must have the file type `'pdvr'`. If the file does not have the correct resource type, QuickDraw GX cannot recognize it as an extension or driver.

The creator type of each printing extension must be unique. The only way to guarantee uniqueness is to register the creator type with Macintosh Developer Technical Support. This is highly recommended because QuickDraw GX relies on the uniqueness of the creator type.

Printing Resources Reference

This section describes the data structures and printing resources that are specific to printing extension and printer driver files.

The "Constants and Data Types" section shows enumerations and structures for some of the resources.

The remaining sections describe the resources used in printing extensions and printer drivers—resources used in both, those used only in printing extensions, and those used only in printer drivers. Most of the resources used for printing extensions and printer drivers have resource templates defined for them. These templates are shown in the illustration that accompanies the section on each resource in this chapter.

Almost all of the fields in the resources are simple, unstructured values. Numerous fields use constant values with specific meanings; the constants that can be used in these fields are shown in the tables that accompany the field descriptions in this section.

The resources that you define for printing extensions and printer drivers are presented in three sections:

- the resources that you use for printing extensions and printer drivers
- the resources that you use only for printing extensions
- the resources that you use only for printer drivers

Some resources required for extensions and drivers are described in other *Inside Macintosh* books. For information on the standard version (`'vers'`), bundle (`'BNDL'`), file reference (`'FREF'`), and various icon resources, all of which are required resources, see *Inside Macintosh: Macintosh Toolbox Essentials*. For information on required resources that define the Chooser interface for your driver, see *Inside Macintosh: Devices*.

If you want to provide a user interface to your printing extension or printer driver, see information on the necessary resources to do this in *Inside Macintosh: QuickDraw GX Printing*.

Constants and Data Types

QuickDraw GX defines structures and enumerations for some of the printing resources. These data types are defined in this section.

The Buffering and Input/Output Preferences Structure

The buffering and input/output preferences structure, of data type `gxIOPrefsRec`, is used to specify the contents of the buffering and input/output preferences (`'iobm'`) resource.

```
struct gxIOPrefsRec {
   unsigned long  communicationsOptions;
   unsigned long  numBuffers;
   unsigned long  bufferSize;
   unsigned long  numReqBlocks;
   unsigned long  openCloseTimeout;
   unsigned long  readWriteTimeout
};

typedef struct gxIOPrefsRec gxIOPrefsRec, *gxIOPrefsPtr,
**gxIOPrefsHdl;
```

Field descriptions

`communicationsOptions`	The options for how a driver handles input and output communications. The value of this field is 0 for standard I/O mechanisms and `gxUseCustomIO` (1) for nonstandard mechanisms, including SCSI.
`numBuffers`	The number of buffers created for the driver.
`bufferSize`	The number of bytes in each buffer.
`numReqBlocks`	The maximum number of input or output requests that can be pending at any time.
`openCloseTimeout`	The number of clock ticks that constitute a timeout when trying to open or close the device.
`readWriteTimeout`	The number of clock ticks that constitute a timeout when trying to read from or write to the device.

The Customization Structure

The customization structure, of data type `gxCustomizationRec`, defines the format of the customization (`'cust'`) resource.

```
struct gxCustomizationRec {
   short horizontalResolution;
   short verticalResolution;
   short upDriverType;
   Point patternStretch;
   short translatorSettings
};

typedef struct gxCustomizationRec gxCustomizationRec,
*gxCustomizationPtr, **gxCustomizationHdl;
```

Field descriptions

`horizontalResolution`	The horizontal resolution to use for the device in dots per inch.
`verticalResolution`	The vertical resolution to use for the device in dots per inch.
`upDriverType`	The Macintosh Printing Manager interface driver with which your device is compatible. The values for this field are shown in Table 6-20 on page 6-48.
`patternStretch`	The scaling factor for printing bitmap patterns.
`translatorSettings`	Settings for translating Macintosh Printing Manager driver calls into messages for your driver. The values for this field are shown in Table 6-21 on page 6-48.

The Resolution Structure

The resolution structure, of data type `gxResolutionRec`, defines the format of the resolution (`'resl'`) resource.

```
struct gxResolutionRec {
   short    rangeType;
   short    xMinimumResolution;
   short    xMaximumResolution;
   short    yMinimumResolution;
   short    yMaximumResolution;
   short    resolutionCount;
   Point    resolutions[1];
};

typedef struct gxResolutionRec gxResolutionRec,
*gxResolutionPtr, **gxResolutionHdl;
```

Field descriptions

`rangeType`	The type of resolution range being defined. This value is currently always 1.
`xMinimumResolution`	The minimum horizontal resolution supported by the driver in dots per inch (dpi).
`xMaximumResolution`	The maximum horizontal resolution supported by the driver in dpi.
`yMinimumResolution`	The minimum vertical resolution supported by the driver in dpi.
`yMaximumResolution`	The maximum vertical resolution supported by the driver in dpi.
`resolutionCount`	The number of entries in the `resolutions` array.
`resolutions`	An array of points, each of which defines a printing resolution, in dots per inch, supported by the driver. The x value of each point defines the horizontal resolution, and the y value of each point defines the vertical resolution.

Raster Preferences Structure

The raster preferences structure, of type `gxRasterPrefsRec`, defines the format of the raster preferences (`'rdip'`) resource.

```
struct  gxRasterPrefsRec{
   gxRasterRenderOptions
                renderOptions;
   Fixed        hImageRes;
   Fixed        vImageRes;
   short        minBandSize;
   short        maxBandSize;
   Fixed        ramPercentage;
   long         ramSlop;
   short        depth;
   short        numPlanes;
   gxPlaneSetupRec
                planeSetup[1];
};

typedef struct gxRasterPrefsRec gxRasterPrefsRec,
*gxRasterPrefsPtr, **gxRasterPrefsHdl;
```

Field descriptions

`renderOptions`	Rendering options for raster imaging. The constants that you can combine into a single value for this field are the values of the raster render options enumeration, described in the next section.
`hImageRes`	The horizontal resolution for imaging.
`vImageRes`	The vertical resolution for imaging.
`minBandSize`	The minimum band size to use, in pixels.
`maxBandSize`	The maximum band size to use, in pixels. A value of 0 in this field indicates that the maximum band size is the full page.
`ramPercentage`	The maximum percentage of available RAM to use.
`ramSlop`	The minimum amount of RAM to leave available.
`depth`	The depth, in pixels, for each plane.
`numPlanes`	The number of planes.
`planeSetup`	An array of structures containing the setup data for each plane.

Raster Render Options

The raster render options enumeration defines constants that you can use in the `renderOptions` field of the raster preferences structure, which is described in the previous section.

```
enum {
   gxDefaultRaster   = 0x00000000,
   gxDontResolveTransferModes
                     = 0x00000001,
   gxRenderInReverse = 0x00000002,
   gxOnePlaneAtATime = 0x00000004,
   gxSendAllBands    = 0x00000008
};

typedef long gxRasterRenderOptions;
```

Constant descriptions

`gxDefaultRaster`	The driver uses the default raster options.
`gxDontResolveTransferModes`	If this is included, the driver does not need to resolve transfer modes. The default is to resolve transfer modes.
`gxRenderInReverse`	The driver needs to traverse the raster image in reverse order.
`gxOnePlaneAtATime`	The driver needs to render each plane separately.
`gxSendAllBands`	The driver needs to send every band of data, even if it is all empty.

Raster Package Structure

The raster package structure, of data type `gxRasterPackageRec`, defines the format of the raster package (`'rpck'`) resource.

```
struct gxRasterPackageRec {
   Ptr      bufferSize;
   short    colorPasses;
   short    headHeight;
   short    numberPasses;
   short    passOffset;
   gxRasterPackgeOptions
            packageOptions;
};

typedef struct gxRasterPackageRec gxRasterPackageRec,
*gxRasterPackagePtr, **gxRasterPackageHdl;
```

Field descriptions

`bufferSize`	The buffer size for packaging. This value must be greater than or equal to the maximum head-pass size.
`colorPasses`	The number of color passes. This value is typically 1 for monochrome printers and 4 for CMYK printers.
`headHeight`	The height of the print head, in pixels.
`numberPasses`	The number of passes it takes to print one print head of data.
`passOffset`	The offset between passes, in pixels.
`packageOptions`	The packaging options. The constants that you can use in this field are the values of the raster package options enumeration, described in the next section. You can combine several of the constants into one value for this field.

Raster Package Options

The raster package options enumeration defines constants that you can use in the `packageOptions` field of the raster package structure, which is described in the previous section.

```
enum {
   gxSendAllColors   = 0x00000001,
   gxInterlaceColor  = 0x00000002,
   gxOverlayColor    = 0x00000004,
   gxUseColor        = (gxInterlaceColor|gxOverlayColor);
};

typedef long gxRasterPackageOptions;
```

Constant descriptions

`gxSendAllColors`	The driver needs to send all bands of data, even when the entire band is empty.
`gxInterlaceColor`	The driver needs to interlace colors because ribbon contamination is a concern on this color printer. If you use this option, you might need to use negative line feed values.
`gxOverlayColor`	The driver does not need to interlace colors because ribbon contamination is not a concern on this printer.
`gxUseColor`	This is a color printer.

Raster Package Controls Structure

The raster package controls structure, of data type `gxRasterPackageControlsRec`, defines the format of the raster package controls (`'ropt'`) resource.

```
struct gxRasterPackageControlsRec {
   short    startPageStringID;
   short    formFeedStringID;
   short    forwardMax;
   gxStandardNumberRec
            forwardLineFeed;
   short    reverseMax;
   gxStandardNumberRec
            reverseLineFeed;
};

typedef struct gxRasterPackageControlsRec
gxRasterPackageControlsRec, *gxRasterPackageControlsPtr,
**gxRasterPackageControlsHdl;
```

Field descriptions

`startPageStringID`	The ID of the wide string (`'wstr'`) to send to the device at the start of each page. Wide strings are strings that require a 16-bit, short integer value to define their length. The length value is stored in the first 2 bytes of the string.
`formFeedStringID`	The ID of the wide string (`'wstr'`) to send to the device to generate a form feed.
`forwardMax`	The maximum amount of a forward line feed.
`forwardLineFeed`	The standard number structure that defines how to express the forward line-feed value.

`reverseMax`	The maximum amount of a reverse line feed.
`reverseLineFeed`	The standard number structure that defines how to express the reverse line-feed value.

Standard Number Structure

The standard numbering structure, of data type `StandardNumberRec`, defines how to output numbers. The raster package controls structure contains two of these structures.

```
struct gxStandardNumberRec {
   short    numberType;
   short    minWidth;
   char     padChar;
   char     alignment;
   Str31    startString;
   Str31    endString;
};

typedef struct gxStandardNumberRec gxStandardNumberRec,
*gxStandardNumberPtr;
```

Field descriptions

`numberType`	The type of numeric output to be used. This is one of the types shown in Table 6-33 on page 6-76.
`minWidth`	If you are using the `RasterNumToASCII` number type, this is the minimum number of characters in the output number string. If you are using the `RasterNumDirect` number type, this is the minimum number of bytes in the output number string.
`padChar`	The character used to pad (in front) numbers that are shorter than the minimum width value.
`alignment`	A single byte used for alignment of data in this structure.
`startString`	The string sent in front of the number string.
`endString`	The string sent after the number string.

Resources Used for Printing Extensions and Printer Drivers

Six resources are used for both printing extensions and printer drivers and are described in this section.

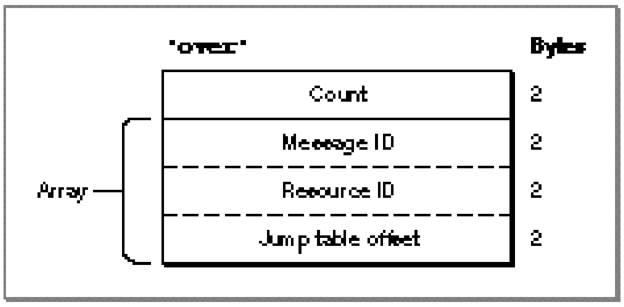
- The override (`'over'`) resource defines which of the printing messages your printing extension or printer driver is overriding. All printing extensions and printer drivers must have this resource.

- The version (`'vers'`) resource defines with which version of QuickDraw GX your printing extension or printer driver is compatible. All printing extensions and printer drivers must have at least one version resource that is used for this purpose.
- The status (`'stat'`) resource defines status messages for display to the user in the desktop printer window during the printing of a job.
- The printing alert (`'plrt'`) resource defines alert messages that are displayed in printing alert boxes when a user's attention is required during the printing of a job.
- The tray count (`'tray'`) resource specifies how many paper trays your printing extension or printer driver supports.
- A tray name (`'tryn'`) resource specifies the name of each paper tray.

The Override ('over') Resource

You must provide at least one override resource, of type `gxOverrideType`, for any printing extension or printer driver that you develop. This resource provides QuickDraw GX with a list of the messages that you are overriding in your extension or driver, along with the ID of the resource in which to find the code for your override's implementation. Figure 6-1 shows the structure of an override resource.

Figure 6-1 The override resource



The override resource consists of a variable number of records, each of which specifies a message, a resource type, a resource ID, and an offset value. Each entry tells QuickDraw GX the name of a printing message that you are overriding and where to find the code for your override.

- Count. The number of message entries in the resource.
- Message ID. The ID of the message that you are overriding. QuickDraw GX defines an integer constant that you use here for each message. These constants are shown in Table 6-2 on page 6-15.
- Resource ID. The ID of the resource in which the code for your override can be found.

- Jump table offset. The offset into the jump table . This is the number of bytes from the beginning of the jump table to the jump instruction for your override code. The first 4 bytes in the jump table must be reserved for use by QuickDraw GX, so your first offset must be 4. You can read about the jump table in the chapter “Printing Extensions” in this book.

You need to provide separate override resources for different kinds of printing messages. There are three kinds of messages:

- Universal messages, used for all imaging systems.
- Imaging system messages, used for specific imaging systems. Some messages are raster imaging messages, some messages are PostScript imaging messages, and some are vector imaging messages.
- Macintosh Printing Manager messages, used to provide compatibility with the Macintosh Printing Manager interface for printing. Printing extensions cannot override Macintosh Printing Manager messages; if an extension does specify that it is overriding a Macintosh Printing Manager call, that message override is ignored.

The ID of each override resource indicates which kind of messages are specified in the resource. Different values are used in override resources for printing extensions than are used for printer drivers, as shown in Table 6-1.

Table 6-1 Override resource IDs

Message type	Extension resource ID	Driver resource ID
Universal	`gxExtensionUniversalOverrideID`	`gxDriverUniversalOverrideID`
Imaging system	`0`	`gxDriverImagingOverrideID`
Macintosh Printing Manager	Not allowed	`gxDriverCompatibilityOverrideID`

Imaging system messages are actually separated into categories: messages for the raster imaging system, PostScript imaging system, and vector imaging system. The identifier used for each message type is shown in Table 6-2.

Table 6-2 Printing message constants

Message type	Constant	Value
Universal	`gxInitialize`	0
	`gxShutDown`	1
	`gxJobIdle`	2
	`gxJobStatus`	3
	`gxPrintingEvent`	4
	`gxJobFormatDialog`	5
	`gxFormatDialog`	6
	`gxJobPrintDialog`	7
	`gxFilterPanelEvent`	8
	`gxHandlePanelEvent`	9
	`gxParsePageRange`	10
	`gxDefaultJob`	11
	`gxDefaultFormat`	12
	`gxDefaultPaperType`	13
	`gxDefaultPrinter`	14
	`gxCreateSpoolFile`	15
	`gxSpoolPage`	16
	`gxSpoolData`	17
	`gxSpoolResource`	18
	`gxCompleteSpoolFile`	19
	`gxCountPages`	20
	`gxDespoolPage`	21
	`gxDespoolData`	22
	`gxDespoolResource`	23
	`gxCloseSpoolFile`	24
	`gxStartJob`	25
	`gxFinishJob`	26
	`gxStartPage`	27
	`gxFinishPage`	28
	`gxPrintPage`	29
	`gxSetupImageData`	30
	`gxImageJob`	31
	`gxImageDocument`	32
	`gxImagePage`	33
	`gxRenderPage`	34
	`gxCreateImageFile`	35
	`gxOpenConnection`	36
	`gxCloseConnection`	37
	`gxStartSendPage`	38
	`gxFinishSendPage`	39
	`gxWriteData`	40
	`gxBufferData`	41
	`gxDumpBuffer`	42
	`gxFreeBuffer`	43

continued

Table 6-2 Printing message constants (continued)

Message type	Constant	Value
	`gxCheckStatus`	44
	`gxGetDeviceStatus`	45
	`gxFetchTaggedData`	46
	`gxGetDTPMenuList`	47
	`gxDTPMenuSelect`	48
	`gxHandleAlertFilter`	49
	`gxJobFormatModeQuery`	50
	`gxWriteStatusToDTPWindow`	51
	`gxInitializeStatusAlert`	52
	`gxHandleAlertStatus`	53
	`gxHandleAlertEvent`	54
	`gxCleanupStartJob`	55
	`gxCleanupStartPage`	56
	`gxCleanupOpenConnection`	57
	`gxCleanupStartSendPage`	58
	`gxDefaultDesktopPrinter`	59
	`gxCaptureOutputDevice`	60
	`gxOpenConnectionRetry`	61
	`gxExamineSpoolFile`	62
	`gxFinishSendPlane`	63
	`gxDoesPaperFit`	64
	`gxChooserMessage`	65
	`gxFindPrinterProfile`	66
	`gxFindFormatProfile`	67
	`gxSetPrinterProfile`	68
	`gxSetFormatProfile`	69
Macintosh Printing Manager	`gxPrOpenDoc`	0
	`gxPrCloseDoc`	1
	`gxPrOpenPage`	2
	`gxPrClosePage`	3
	`gxPrintDefault`	4
	`gxPrStlDialog`	5
	`gxPrJobDialog`	6
	`gxPrStlInit`	7
	`gxPrJobInit`	8
	`gxPrDlgMain`	9
	`gxPrValidate`	10
	`gxPrJobMerge`	11
	`gxPrGeneral`	12
	`gxConvertPrintRecordTo`	13
	`gxConvertPrintRecordFrom`	14
	`gxPrintRecordToJob`	15
Raster imaging system	`gxRasterDataIn`	0
	`gxRasterLineFeed`	1
	`gxRasterPackageBitmap`	2

Table 6-2 Printing message constants (continued)

Message type	Constant	Value
PostScript imaging system	`gxPostscriptQueryPrinter`	0
	`gxPostscriptInitializePrinter`	1
	`gxPostscriptResetPrinter`	2
	`gxPostscriptExitServer`	3
	`gxPostscriptGetStatusText`	4
	`gxPostscriptGetPrinterText`	5
	`gxPostscriptScanStatusText`	6
	`gxPostscriptScanPrinterText`	7
	`gxPostscriptGetDocumentProcSetList`	8
	`gxPostscriptDownloadProcSetList`	9
	`gxPostscriptGetPrinterGlyphsInformation`	10
	`gxPostscriptStreamFont`	11
	`gxPostscriptDoDocumentHeader`	12
	`gxPostscriptDoDocumentSetUp`	13
	`gxPostscriptDoDocumentTrailer`	14
	`gxPostscriptDoPageSetUp`	15
	`gxPostscriptSelectPaperType`	16
	`gxPostscriptDoPageTrailer`	17
	`gxPostscriptEjectPage`	18
	`gxPostscriptProcessShape`	19
Vector imaging system	`gxVectorPackageShape`	0
	`gxVectorLoadPens`	1
	`gxVectorVectorizeShape`	2

Listing 6-1 shows a typical override resource definition. You can also find examples of override resource definitions in Listing 2-24 on page 2-38 in the chapter “Printing Extensions” and in Listing 3-20 on page 3-56 in the chapter “Printer Drivers.”

Listing 6-1 An example of an override resource

```
#define mySegment 0
#define firstOffset 4

resource gxOverrideType (gxExtensionUniversalOverrideID, sysHeap,
                                                       purgeable)
{
   {
   /* message to override     segmentID    jump table offset */

      gxInitialize,           mySegment,     firstOffset+0,
      gxShutDown,             mySegment,     firstOffset+4,
      gxJobPrintDialog,       mySegment,     firstOffset+8,
      gxHandlePanelEvent,     mySegment,     firstOffset+12,
      gxDespoolPage,           mySegment,     firstOffset+16
   };
};
```

This resource specifies the universal messages that an extension is overriding. In this case, five messages are being overridden. Each of these messages is found in a code segment with a resource ID of `mySegment`. The jump (`'JMP'`) statement for the `gxInitialize` message is found at the first offset (4 bytes) from the start of the jump table, and the jump statements for the subsequent messages are each located 4 bytes apart.

IMPORTANT

Each jump table entry is 4 bytes long, and QuickDraw GX reserves the first 4 bytes in the table for its own use. This means that your first jump table entry must be located at an offset of 4 bytes from the beginning of the table. For this reason, the `firstOffset` constant in Listing 6-1 is defined with a value of 4. s

The Version ('vers') Resource

You need to include a version resource that defines with which version of QuickDraw GX your printing extension or printer driver is compatible. In the resource files for your printing extension, you can also include additional standard version resources, as described in *Inside Macintosh: Macintosh Toolbox Essentials*.

For printing extensions, you need to include a version resource with an ID of `gxPrintingExtensionBaseID` that defines with which version of QuickDraw GX your extension is compatible. For printer drivers, you need to include a version resource with an ID of `gxPrintingDriverBaseID` that defines with which version of QuickDraw GX your driver is compatible.

The first byte defines the QuickDraw GX version. For the current version of QuickDraw GX, the first byte must have a value of either 1 or 0. Listing 6-2 shows the version resources that define QuickDraw GX compatibility for the background picture printing extension and for the ImageWriter II printer driver.

Listing 6-2 QuickDraw GX version resources

```
resource 'vers' (gxPrintingExtensionBaseID, sysHeap, purgeable) {
   0x0,
   0x0,
   release,
   0x0,
   verUS,
   "",
   ""
};
```

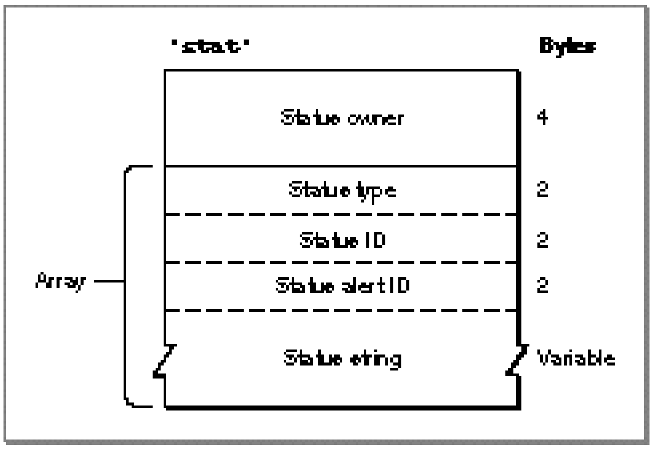
```
resource 'vers' (gxPrintingDriverBaseID, sysHeap, purgeable)
{
   0x01, 0x00, release, 0x00,
   verUS,
   "1.00",
   "1.00, Copyright \251 Apple Computer, Inc. 1989-1993"
};
```

The Status ('stat') Resource

You need to include a status resource, of type `statusType`, to define the status messages that are displayed during the printing process. The driver forces status messages to be displayed by calling either the `GXReportStatus` function (for simple messages) or the `GXAlertTheUser` function (for messages that require user attention).

Figure 6-2 shows the structure of the status resource.

Figure 6-2 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 6-3.
- 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.
- 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 6-3 shows the status type constants and the side effects associated with each.

Table 6-3 Status types

Constant	Value	Explanation of side effects
`gxNonFatalError`	1	Affects the icon that is displayed during spooling
`gxFatalError`	2	Displays a printing alert during spooling
`gxPrinterReady`	3	Signals that alert mode is done
`gxUserAttention`	4	Signals initiation of a modal alert
`gxUserAlert`	5	Signals initiation of a printing alert
`gxPageTransmission`	6	Signals that a page has been sent to the printer and increments the printed page count
`gxOpenConnectionStatus`	7	Signals that animation of the printer icon is to begin
`gxInformationalStatus`	8	Displays an informational status message with no side effects
`gxSpoolingPageStatus`	9	Signals that a page has been spooled and increments the spooled page count
`gxEndStatus`	10	Signals the end of spooling
`gxPercentageStatus`	11	Signals the percentage of the current print job that is currently complete

Listing 6-3 shows an example of a status resource.

Listing 6-3 An example of a status resource

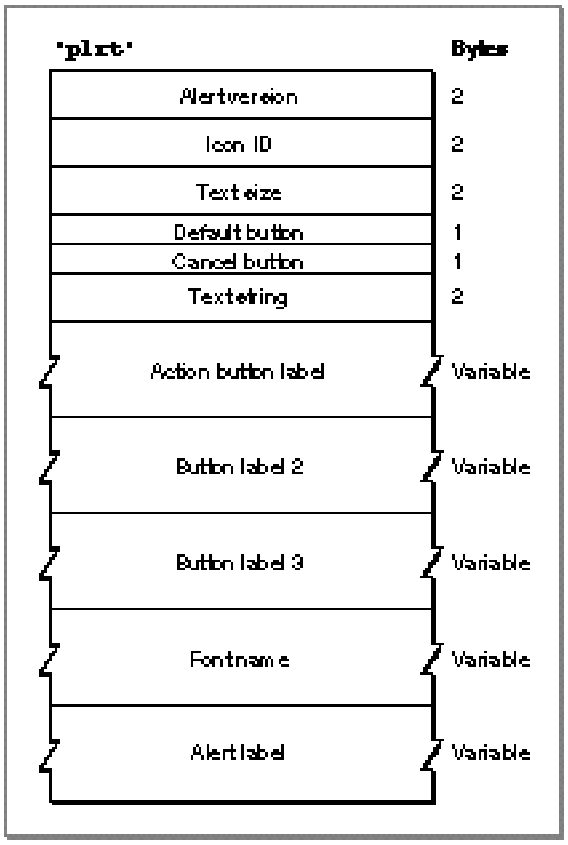
```
#define kDrvrCreatorType 'IWII'

resource statusType (kDriverStatus, sysHeap, purgeable)
{
   kDrvrCreatorType,
   {
   gxInformationalStatus, 1, 0, "Sending data to printer";
   gxUserAlert, 1, kDriverStatus,
                    "Please check that the printer is on-line";
   };
};
```

Listing 6-3 defines a status resource for the ImageWriter II printer driver. This resource defines two status text strings. The first text string is an informational message that does not require a printing alert box and is displayed in the desktop printer window. The second text string requires user attention and is displayed in a printing alert box.

The Printing Alert ('plrt') Resource

You need to include a printing alert resource to define the messages that require user attention during the printing process. Figure 6-3 shows the structure of the printing alert resource.

Figure 6-3 The printing alert resource



The printing alert resource contains a number of fields that you can use to define alert messages for display to the user in printing alert boxes.

- Alert version. The type of printing alert that this resource represents. There are currently 2 versions, as shown in Table 6-4.
- Icon ID. The type of icon to display in the upper-left corner of the alert. The icon types are shown in Table 6-5.
- Text size. The size of the text to display in the alert. You can use the value `defaultSystemSize` to indicate that the default text size for the system is to be used.
- Default button. The string to display on the default button. You use one of the values shown in Table 6-7.
- Cancel button. The string to display on the cancel button. You use one of the values shown in Table 6-7.
- Text string. The string to display in the alert window.

- Action button label. The string that is displayed on the action button. This is a required value.
- Button label 2. The string that is displayed on button 2. This is an optional value.
- Button label 3. The string that is displayed on button 3. This is an optional value.
- Font name. The name of the font used to display the text in this alert.
- Alert title. The title displayed for this alert.

Table 6-4 shows the constants that you can use to specify which kind of printing alert you are defining.

Table 6-4 Printing alert versions

Constant	Value	Explanation
`printingAlert`	1	Produces a modal alert dialog box
`printingStatus`	2	Produces a printing alert box that can only be used with QuickDraw GX printing

Table 6-5 shows the constants that you can use to specify the icon ID for a printing alert resource.

Table 6-5 Icon IDs for a printing alert resource

Constant	Value	Explanation
`noIcon`	–1	No icon is displayed
`stopIcon`	0	The stop icon is displayed
`noteIcon`	1	The note icon is displayed
`cautionIcon`	2	The caution icon is displayed

Table 6-7 shows the constants that you can use to specify the default button strings in a printing alert resource.

Table 6-6 Default button string values for a printing alert resource

Constant	Value	Explanation
`noDefaultTitle`	0	There is no default button
`defaultAction`	1	Use the action button label string
`defaultTitle2`	2	Use the button label 2 string
`defaultTitle3`	3	Use the button label 3 string

Table 6-7 shows the constants that you can use to specify the cancel button strings in a printing alert resource.

Table 6-7 Cancel button string values for a printing alert resource

Constant	Value	Explanation
`noCancelTitle`	0	There is no cancel button
`cancelAction`	1	Use the action button label string
`cancelTitle2`	2	Use the button label 2 string
`cancelTitle3`	3	Use the button label 3 string

Listing 6-4 shows an example of a printing alert resource.

Listing 6-4 An example of a printing alert resource

```
resource 'plrt' (kDriverStatus, sysHeap, purgeable)
{
   printingStatus,
   cautionIcon,
   defaultSystemSize,
   defaultAction,
   noCancelTitle,
   "The document “!1” cannot be printed, because the printer "
   "“!0” is offline.  To continue printing, please make "
   "sure the printer is "
   "properly connected and turned on.  If you wish "
   "to cancel printing, please click Cancel Printing.",
   "Cancel Printing",
   "",
   "",
   " ",
   "Printer offline"
};
```

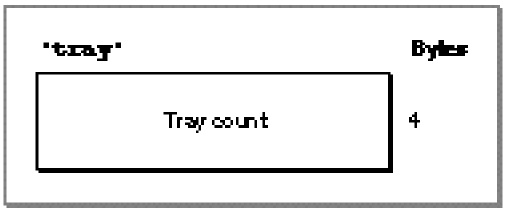
This resource defines the contents of the printing alert box that corresponds to the second entry in the status resource that is shown in Listing 6-3 on page 6-21. The printing alert box contains a caution icon and a Cancel button. The text is displayed using the default system font and text size.

The Tray Count ('tray') Resource

You need to include a tray count resource, of type `gxTrayCountDataType`, if you support named paper trays in your printing extension or printer driver. You use the tray count resource to specify how many paper trays there are on the output device, and you provide a tray name resource (as described in the next section) for each of the trays.

Figure 6-4 shows the structure of the tray count resource.

Figure 6-4 The tray count resource



The tray count resource contains a single value.

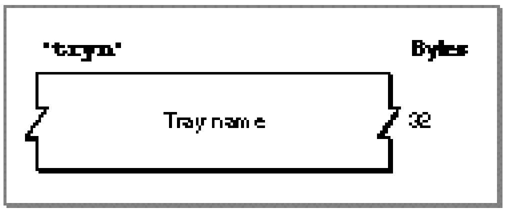
- Tray count. The number of paper trays that your printing extension or printer driver supports on the output device. This is a long integer value.

The Tray Name ('tryn') Resource

You need to include a tray name resource, of type `gxTrayNameDataType`, for each paper tray that your printing extension or printer driver is supporting. You must specify the number of the paper tray as the ID of the resource.

Figure 6-5 shows the structure of the tray name resource.

Figure 6-5 The tray name resource



The tray name resource contains a single value.

- Tray name. The name of the paper tray. This is a Pascal string with a length byte and 31 characters.

Resources Used Only for Printing Extensions

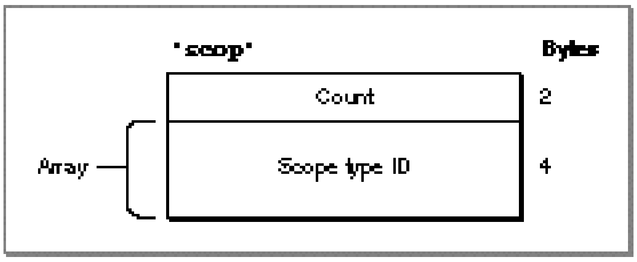
Some resources are used for printing extensions, but not for drivers. These resources are described in this section. They are required for all printing extensions.

- The extension scope (`'scop'`) resource specifies with which imaging systems and drivers your extension is compatible.
- The extension load (`'load'`) resource tells QuickDraw GX where to load your extension in the print message handler chain.
- The extension optimization (`'eopt'`) resource provides QuickDraw GX with information about when your extension needs to be in memory, which allows QuickDraw GX to optimize printing performance.

The Extension Scope ('scop') Resource

You must define at least one extension scope resource, of type `gxExtensionScopeType`, for each printing extension that you develop. This resource indicates the types of drivers and imaging systems with which your extension is compatible. Figure 6-6 shows the structure of an extension scope resource.

Figure 6-6 The extension scope resource



The extension scope resource consists of a variable number of scope values.

- Count. The number of scope types defined in this resource.

- Scope type ID. A long integer value that specifies an imaging system or driver. These values are specified as four-character resource-type names in your resource definition. Constants that you can use for the resource types are shown in Table 6-8.

Table 6-8 Imaging system identifiers for the extension scope resource

Constant	Value	Explanation
`gxAnyPrinterType`	`'univ'`	All output devices
`gxRasterPrinterType`	`'rast'`	Raster output devices
`gxPostscriptPrinterType`	`'post'`	PostScript output devices
`gxVectorPrinterType`	`'vect'`	Vector output devices

The ID of each extension scope resource defines the type of information that is contained in the resource values. Constants for the scope resource IDs are shown in Table 6-9.

Table 6-9 Scope resource identifiers

Constant	Value	Explanation
`gxDriverScopeID`	`gxPrintingExtensionBaseID`	The imaging system types supported by the extension
`gxPrinterScopeID`	`gxPrintingExtensionBaseID+1`	Specific driver types supported by the extension
`gxPrinterExceptionScopeID`	`gxPrintingExtensionBaseID+2`	Specific driver types not supported by the extension

You can include several different extension scope resources in your extension file to pinpoint the scope of your extension. The values in all scope resources are unioned together to produce the exact scope of your extension. A typical pair of scope resources for an extension is shown in Listing 6-5. You can find other examples of scope resources in Listing 2-21 on page 2-35 and Listing 2-22 on page 2-36 in the chapter “Printing Extensions.”

Listing 6-5 An example of a pair of extension resources used together

```
resource gxExtensionScopeType (gxPrinterScopeID, sysHeap,
                                                    purgeable)
{
   {
      gxPostscriptPrinterType,
      gxRasterPrinterType
   };
};

resource gxExtensionScopeType (gxPrinterExceptionScopeID, sysHeap,
                                                        purgeable)
{
   {
      'ODD1'
   };
};
```

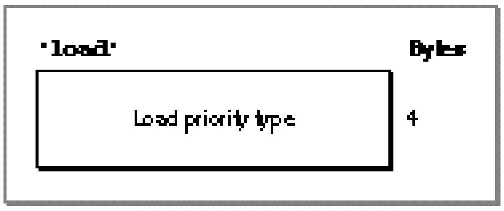
In this example, the first extension scope resource defines the extension as compatible with all PostScript and raster printers. The second resource specifies that the extension is not compatible with the printer driver name `'ODD1'`. The net result is that the extension is compatible with all PostScript and raster printers that do not use the `'ODD1'` driver.

The Extension Load ('load') Resource

The extension load resource, of type `gxExtensionLoadType`, tells QuickDraw GX where to load an extension into the printing message chain. You must include a load resource in your extension file. The value that you define in your extension load resource specifies the default loading order for your extension when it is first added to the system. The user can modify the loading order of extensions by using the Extension Setup dialog box for each desktop printer.

Figure 6-7 shows the structure of an extension load resource.

Figure 6-7 The extension load resource



The extension load resource consists of a single value.

- Load priority type. The default location in the message chain where your extension is to be loaded. Constants for the three load priority types are shown inTable 6-10.

Table 6-10 Extension load priority constants

Constant	Value	Explanation
`gxExtensionLoadFirst`	0x00000100	Load this extension as the first handler in the print message handling chain so that it has the first opportunity to respond to messages that it overrides
`gxExtensionLoadAnywhere`	0x7FFFFFFF	Load this extension anywhere in the print message handling chain
`gxExtensionLoadLast`	0xFFFFFF00	Load this extension last in the print message handling chain so that it receives its message after any other handlers have performed their operations

The extension load resource ID needs to be the constant `gxExtensionLoadID`. Listing 6-6 shows an example of an extension load resource.

Listing 6-6 An example of an extension load resource

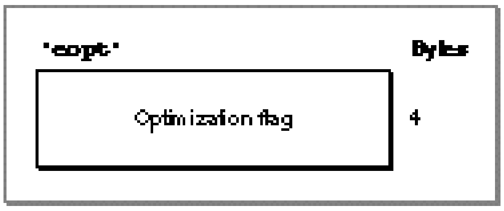
```
resource gxExtensionLoadType (gxExtensionLoadID, sysHeap,
                                                 purgeable)
{
   gxExtensionLoadLast
};
```

This resource tells QuickDraw GX to load the extension last in the message chain, which allows the extension to modify the contents of a page after any other message handlers have performed their modifications. However, the user can change this by using the Extension Setup dialog box for each desktop printer.

The Extension Optimization ('eopt') Resource

The extension optimization resource, of type `gxExtensionOptimizationType`, provides QuickDraw GX with additional information about the activities of an extension. The information in this resource allows QuickDraw GX to optimize its use. You must include an extension optimization resource in each printing extension. Figure 6-8 shows the structure of an extension optimization resource.

Figure 6-8 The extension optimization resource



- Optimization flag. The flags that define when your extension needs to be loaded into memory. Each extension optimization resource contains one or more optimization flags. The constants for these flags are shown in Table 6-11. The flags actually come in pairs: one that specifies that an extension needs to be activated during a certain phase of printing, and a complementary one that specifies that the extension does not need to be active during that phase. QuickDraw GX defines each flag as a Boolean value.

Table 6-11 Extension optimization resource flags

Constant	Value	Explanation
`gxExecuteDuringImaging`	`true`	Use if your extension overrides any messages that are sent during the imaging phase of printing.
`gxDontExecuteDuringImaging`	`false`	Use if your extension only overrides messages that are sent for working with dialog boxes or during the spooling phase of printing.
`gxNeedDeviceStatus`	`true`	Use if your extension performs any form of two-way communications with the device. This applies mostly to extensions driving add-on devices.
`gxDontNeedDeviceStatus`	`false`	Use if your extension does not need to perform two-way communications.
`gxChangePageAtGXDespoolPage`	`true`	Use if your extension changes the contents of a page during processing of an override for the `GXDespoolPage` message.
`gxDontChangePageAtGXDespoolPage`	`false`	Use if your extension does not change the contents of a page during processing of the `GXDespoolPage` message.
`gxChangePageAtGXImagePage`	`true`	Use if your extension changes the contents of a page during processing of an override for the `GXImagePage` message.
`gxDontChangePageAtGXImagePage`	`false`	Use if your extension does not change the contents of a page during processing of the `GXImagePage` message.

Table 6-11 Extension optimization resource flags (continued)

Constant	Value	Explanation
`gxChangePageAtGXRenderPage`	`true`	Use if your extension changes the contents of a page during processing of an override for the `GXRenderPage` message.
`gxDontChangePageAtGXRenderPage`	`false`	Use if your extension does not change the contents of a page during processing of the `GXRenderPage` message.
`gxServerPresenceRequired`	`true`	Use if your extension must be available on the print server's system.
`gxNotServerPresenceRequired`	`false`	Use if your extension does not have to be available on the print server's system.
`gxClientPresenceRequired`	`true`	Use if your extension must be available on the user's system.
`gxNotClientPresenceRequired`	`false`	Use if your extension does not have to be available on the user's system.

IMPORTANT

You need to be careful about setting these flags appropriately. Unnecessarily setting these flags can cause performance degradation during printing to certain devices. s

The extension optimization resource ID needs to be the constant `gxExtensionOptimizationID`. Listing 6-7 shows an example of an extension optimization resource. You can also find an example of an extension optimization resource definition in Listing 2-23 on page 2-37 in the chapter "Printing Extensions."

Listing 6-7 An example of an extension optimization resource

```
resource gxExtensionOptimizationType (gxExtensionOptimizationID,
                                        sysHeap, purgeable)
{
   gxDontExecuteDuringImaging,
   gxDontNeedDeviceStatus,
   gxChangePageAtGXDespoolPage,
   gxDontChangePageAtGXImagePage,
   gxDontChangePageAtGXRenderPage,
   gxNotServerPresenceRequired,
   gxClientPresenceRequired
};
```

This example specifies that the extension does not have to be available during the imaging phase of printing, does not need to perform two-way communications, and only changes the page during the despooling process of the imaging phase of printing.

This extension does not need to be present on the server, but must be present on the client machine.

Resources Used Only in Printer Drivers

Some resources are used for printer drivers, but not for extensions. These resources, which are summarized in Table 6-12, are described in this section.

Table 6-12 Resources used only in printer drivers

Resource	Function	Status
Imaging system (`'isys'`)	Identifies the imaging system that the driver uses.	Required
Look (`'look'`)	Specifies the kinds of communications that a desktop printer uses.	Required
Communications (`'comm'`)	Specifies the parameters of device communications that are used by the driver.	Required
Customization (`'cust'`)	Defines translation parameters for compatibility with the Macintosh Printing Manager.	Optional (used for compatibility)
Resolution (`'resl'`)	Defines the horizontal and vertical resolution values supported by a driver for compatibility with the Macintosh Printing Manager.	Optional (used for compatibility)
Print record (`'PREC'`)	Defines a default print record for driver compatiblity with the Macintosh Printing Manager.	Optional (used for compatibility)
Dialog control (`'dctl'`)	Automates the definition of control items in dialog boxes that are used for compatibility with the Macintosh Printing Manager.	Optional (used for compatibility)
Scaling table (`'stab'`)	Defines the discrete page size reduction values that a user can select when printing with the Macintosh Printing Manager.	Optional (used for compatibility)
Buffering (`'iobm'`)	Defines the timeout and buffering parameters for a driver.	Optional
Capture (`'cpts'`)	Defines the strings that allow a device to be removed and replaced on a network during printer sharing.	Optional

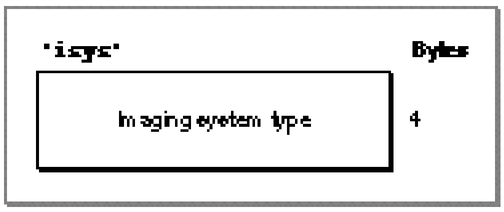
Table 6-12 Resources used only in printer drivers (continued)

Resource	Function	Status
Color set ('crst')	Defines color information for a raster printer driver.	Optional
Raster preferences ('rdip')	Defines rendering options for a raster driver.	Required for raster drivers
Raster package ('rpck')	Defines how bitmap data is packed into rasters for raster drivers.	Optional
Raster package controls ('ropt')	Defines how some forms of line feeding are performed on a raster device.	Optional
PostScript scanning ('scan')	Specifies a set of replacement strings that a printer driver uses when interpreting the information returned by a PostScript printer.	Optional
PostScript procset ('prec')	Defines a PostScript procedure set that can be downloaded to an output device.	Optional
PostScript font type ('pfnt')	Defines parameters for a specific font type used on a PostScript printer.	Optional
PostScript preferences ('pdip')	Defines rendering options for a PostScript printer driver.	Required for PostScript drivers

The Imaging System ('isys') Resource

The imaging system resource, of type `gxImagingSystemSelectorType`, tells QuickDraw GX which imaging system a driver uses. You must include an imaging system resource in your driver file. Figure 6-9 shows the structure of an imaging system resource, which consists of one value.

Figure 6-9 The imaging system resource



- Imaging system type. The imaging system with which your driver operates. This value is specified in your resource files as a four-character imaging system name, for which you can use the imaging system constants that are shown in Table 6-8 on page 6-27. Note that you cannot specify the `gxAnyPrinterType` value in this resource.

The imaging system resource ID needs to be the constant `gxImagingSystemSelectorID`. Listing 6-8 shows an example of an imaging system resource.

Listing 6-8 An example of an imaging system resource

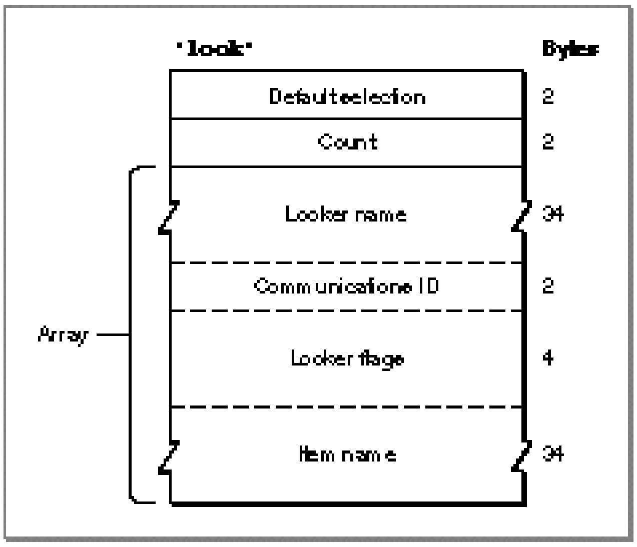
```
resource gxImagingSystemSelectorType (gxImagingSystemSelectorID,
                                              sysHeap, purgeable)
{
   gxRasterPrinterType
};
```

This resource tells QuickDraw GX that the driver uses the raster imaging system.

The Look ('look') Resource

The look (`'look'`) resource specifies the kinds of communications that a driver uses. It contains a list of entries, each of which can be used when the user creates a desktop printer with the Chooser. The values in the look resource are also used for printer sharing; you must include a communications resource for each entry in the look resource to make printer sharing work properly. Figure 6-10 shows the structure of a look resource.

Figure 6-10 The look resource



The look resource consists of a default indicator, a count, and one or more "looker" entries.

- Default selection. A value that specifies which looker entry is the default.
- Count. The number of looker entries in this resource.

Each looker entry tells QuickDraw GX about one kind of communications that the driver can perform and includes the following fields:

- Looker name. The name to display in the list of communications choices that are presented to the user.
- Communications ID. The resource ID of the communications (`'comm'`) resource that corresponds to this looker entry.
- Looker flags. A collection of flags that can be combined to define what kind of connection the entry is naming. The flag constants shown are in Table 6-13.
- Item name. The name that is displayed for the item in the Chooser. For serial connections, this is the default port name. For AppleTalk and PrinterShare connections, this is the name-binding protocol (NBP) type.

Table 6-13 shows the constants that you can use to specify the kind of connection that a driver uses.

Table 6-13 Flag constants for the look resource

Constant	Value	Explanation
`isAppleTalk`	1	This entry specifies an AppleTalk Printer Access Protocol (PAP) connection that requires a name-binding proctocol (NBP) lookup.
`iconCells`	2	This entry specifies that the Chooser displays icons in its list rather than names.
`isPrinterShare`	4	This entry specifies an Apple PrinterShare connection to a server.

The look resource ID needs to be the constant –4096. Listing 6-9 shows an example of a look resource.

Listing 6-9 An example of a look resource

```
resource 'look' (-4096, sysHeap, purgeable)
{
   2,                /* use the 2nd entry in list as default */
   {
   “AppleTalk”,   -4096,   isAppleTalk,      “ImageWriter”;
   “Serial”,      -4095,   iconCells,        “Modem Port”;
   “Servers”,     -4094,   isAppleTalk+isPrinterShare,
                                             “ImageWriterIIIS”;
   };
};
```

This is a look resource for an ImageWriter II printer, which supports PAP (the Printer Access Protocol used in AppleTalk), serial, and PrinterShare communications interfaces. The communications resource that defines the AppleTalk interface has resource ID –4096, the communications resource for the serial interface has resource ID –4095, and the communications resource for the PrinterShare interface has resource ID –4094. The default connection type is serial, the Chooser uses icons for choosing the serial connection port, and the default icon that would be selected by the Chooser is the “Modem Port” icon.

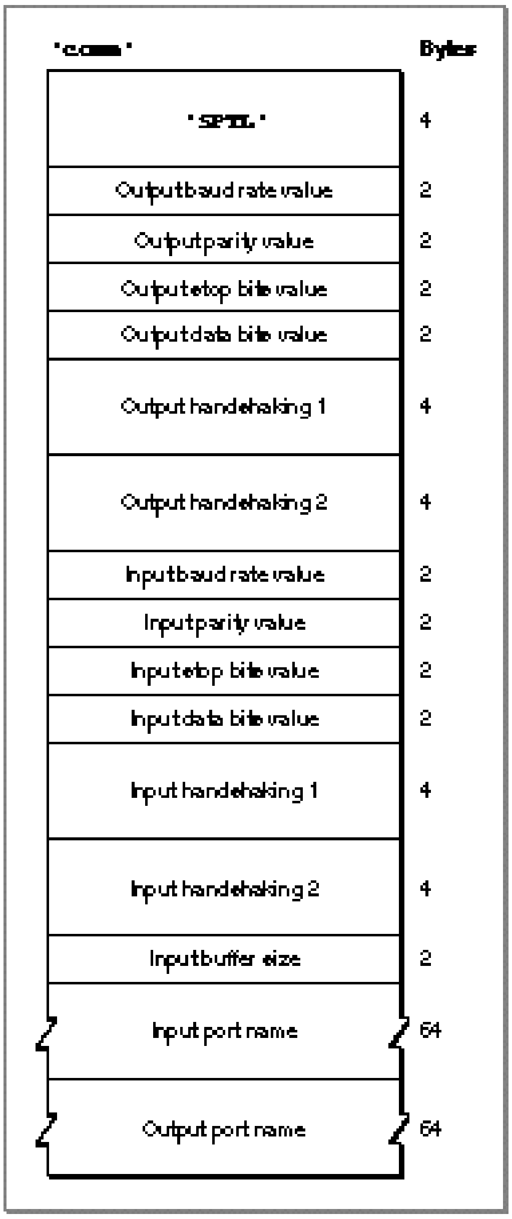
The Communications ('comm') Resource

Each communications (`'comm'`) resource, of type `gxDeviceCommunicationsType`, defines parameters for a specific communications protocol that is supported by a printer driver. You must include one communications resource for each entry in your look resource. You can define printer communications resources in four different formats: serial communications resources, PAP communications resources, PrinterShare communications resources, and SCSI communications resources.

Serial Communications Resource

You need to define a serial communications resource to support serial communications in your driver. Figure 6-11 shows the structure of this form of a communications resource.

Figure 6-11 The serial communications resource



The serial communications resource consists of a communications type identifier and a number of input and output communications values.

- Type identifier. `'SPTL'` (or the constant `Serial`). Used to identify serial communications resources.
- Output baud-rate value. The baud rate used for output, constants for which are shown in Table 6-14.

- Output parity value. The parity type used for output, constants for which are shown in Table 6-15.
- Output stop-bits value. The number of stop bits used for output, constants for which are shown in Table 6-16.
- Output data-bits value. The number of data bits used for output, constants for which are shown in Table 6-17.
- Output handshaking. The handshaking protocol used for output, as defined on page 6-40.
- Input baud-rate value. The input baud rate, constants for which are shown in Table 6-14.
- Input parity value. The parity type used for input, constants for which are shown in Table 6-15.
- Input stop-bits value. The number of stop bits used for input, constants for which are shown in Table 6-16.
- Input data-bits value. The number of data bits used for input, constants for which are shown in Table 6-17.
- Input handshaking. The handshaking protocol used for input, as defined on page 6-40.
- Input buffer size. The number of bytes in the serial input buffer.
- Input port name. The name of the input port, filled in by the Chooser.
- Output port name. The name of the output port, filled in by the Chooser.

Table 6-14 shows the values that you can use to specify the baud rate in serial communications resources.

Table 6-14 Constants for the baud rate

Constant	Explanation
`baud300`	300 baud
`baud600`	600 baud
`baud1200`	1200 baud
`baud1800`	1800 baud
`baud2400`	2400 baud
`baud3600`	3600 baud
`baud4800`	4800 baud
`baud7200`	7200 baud
`baud9600`	9600 baud
`baud19200`	19200 baud
`baud38400`	38400 baud
`baud57600`	57600 baud

Table 6-15 shows the constants that you can use to specify parity values in serial communications resources.

Table 6-15 Parity constants

Constant	Explanation
`noParity`	Parity not used
`oddParity`	Use odd parity
`evenParity`	Use even parity

Table 6-16 shows the constants that you can use to specify stop-bits values in serial communications resources.

Table 6-16 Constants for stop bits

Constant	Explanation
`oneStop`	1
`oneFiveStop`	1.5
`twoStop`	2

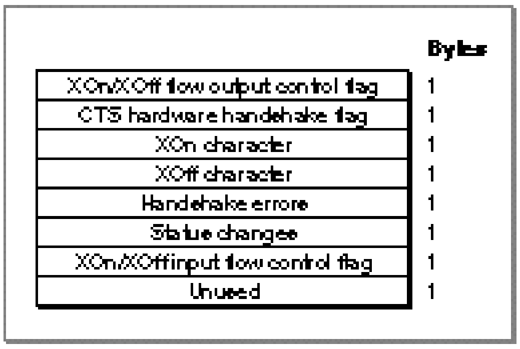
Table 6-17 shows the constants that you can use to specify data-bits values in serial communications resources.

Table 6-17 Constants for data bits

Constant	Explanation
`data5`	5 data bits
`data6`	6 data bits
`data7`	7 data bits
`data8`	8 data bits

The values that are used to define input and output handshaking in the serial communications resource are shown in Figure 6-12.

Figure 6-12 Input and output handshaking values


Bytes
XOn/XOff flow output control flag
1
CTS hardware handshake flag
1
XOn character
1
XOff character
1
Handshake errors
1
Status changes
1
XOn/XOff input flow control flag
1
Unused
1

The handshake values define parameters of how the driver and the device communicate.

- XOn / XOff output flow-control flag. A value that defines whether output flow control is enabled or disabled. If this value is nonzero, XOn / XOff output flow is enabled.
- CTS hardware handshake flag. A value that defines whether CTS hardware handshake is enabled or disabled. If this value is nonzero, CTS hardware handshake is enabled.
- XOn character. The XOn character used for XOn / XOff flow control.
- XOff character. The XOff character used for XOn / XOff flow control.
- Handshake errors. The types of errors that cause input requests to be aborted. The constants that you can use for the error types are shown in Table 6-18. These constants can be combined into a single value to specify all of the error types that cause input requests to be aborted.
- Status changes. A value that indicates whether changes in the CTS or break status cause the Serial Driver to post device-driver events. The constants that you can use for the status event types are shown in Table 6-18. These constants can be combined into a single value to specify all of the status changes that cause device driver events to be posted.

  Use of the status change option is discouraged because interrupts will be disabled for a long time while the driver event is posted.
- XOn / XOff input flow-control flag. A value that defines whether input flow control is enabled or disabled. If this value is nonzero, XOn / XOff input flow is enabled.
- Unused. A single byte reserved for use in the future.

Table 6-18 shows the constants that you can use to specify handshake error-handling in serial communications resources.

Table 6-18 Handshake error types

Constant	Value	Explanation
`parityErr`	16	Abort the input request if a parity error occurs
`hwOverrunErr`	32	Abort the input request if a hardware overrun error occurs
`framingErr`	64	Abort the input request if a framing error occurs

Table 6-19 shows the constants that you can use to specify which status changes cause driver events to be posted.

Table 6-19 Status changes that cause driver events to be posted

Constant	Value	Explanation
`ctsEvent`	32	Causes driver events to be posted if the CTS status changes
`breakEvent`	128	Causes driver events to be posted if the break status changes

Listing 6-10 shows an example of a serial communications resource.

Listing 6-10 An example of a serial communications resource

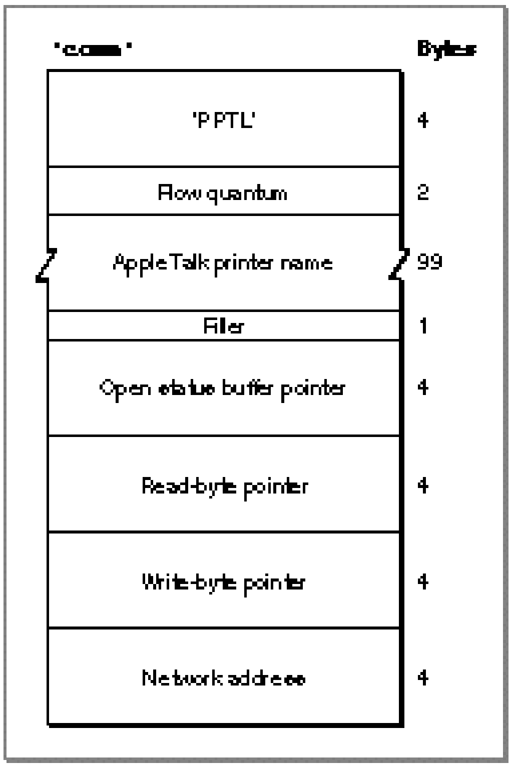
```
resource gxDeviceCommunicationsType (-4095, sysHeap, purgeable)
{
   Serial
      {
      baud9600,      /* output baud rate */
      noParity,      /* output parity */
      oneStop,       /* output stop bits */
      data8,         /* output data size */
      0x00010000,    /* output handshaking */
      0x00000000,
      baud9600,      /* input baud rate */
      noParity,      /* input parity */
      oneStop,       /* input stop bits */
      data8,         /* input data size */
      0,             /* input handshaking */
      0,
      1024,          /* input buffer size */
      ".Ain",        /* input driver name; Chooser fills in */
      ".Aout"        /* output driver name; Chooser fills in  */
      };
};
```

This example defines a serial port that supports input and output using the same parameters: 9600 baud, no parity, 1 stop bit, and 8 data bits. The input buffer size is 1024 bytes long.

AppleTalk PAP Communications Resource

You need to define a PAP communications resource to support the AppleTalk Printer Access Protocol in your driver. Figure 6-13 shows the structure of this form of a communications resource.

Figure 6-13 The PAP communications resource



The PAP communications resource consists of a communications type identifier and a number of input and output communications values.

- Type identifier. 'PPTL'(or the constant PAP). Used to identify PAP communications resources.
- Flow quantum. A value that specifies the flow quantum, which is the number of packets that your device sends or receives in one transaction. The flow quantum value is hardware dependent: it is defined by the hardware manufacturer. The flow quantum value for LaserWriter printers is 8, the value for ImageWriters is 1. You can read more about this value in *Inside AppleTalk*.
- AppleTalk printer name. A string value that is filled in by the Chooser with the compacted, network name of the printer.
- Filler. An unused byte added for alignment purposes.
- Open status buffer pointer. An internally used value. This must be 0 in your resource definition.
- Read-byte pointer. An internally used value. This must be 0 in your resource definition.
- Write-byte pointer. An internally used value. This must be 0 in your resource definition.
- Network address. An internally used value. This must be 0 in your resource definition.

Listing 6-11 shows an example of a PAP communications resource that defines an AppleTalk connection for an ImageWriter II printer, which uses a flow quantum value of 1.

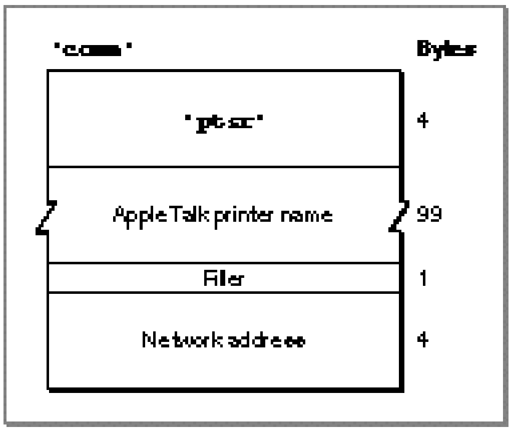
Listing 6-11 An example of an AppleTalk communications resource

```
resource gxDeviceCommunicationsType(-4096, sysHeap, purgeable)
{
   PAP
      {
      1,            /* flow quantum */
      ""            /* AppleTalk address; Chooser fills in */
      0, 0, 0, 0
      };
};
```

PrinterShare Communications Resource

You need to define a PrinterShare communications reource to support a server connection to your driver. Figure 6-14 shows the structure of this form of a communications resource.

Figure 6-14 The PrinterShare communications resource



The PrinterShare communications resource consists of a communications type identifier and a number of input and output communications values.

- Type identifier. `'ptsr'`(or the constant `PrinterShare`). Used to identify PrinterShare communications resources.
- AppleTalk printer name. A string value filled in by the Chooser with the compacted, network name of the printer.
- Filler. An unused byte added for alignment purposes.
- Network address. The most recent network address of the printer. This value is filled in at run time and needs to be 0 in your resource definition.

Listing 6-12 shows an example of a PrinterShare communications resource that defines a print server's connection for an ImageWriter II printer.

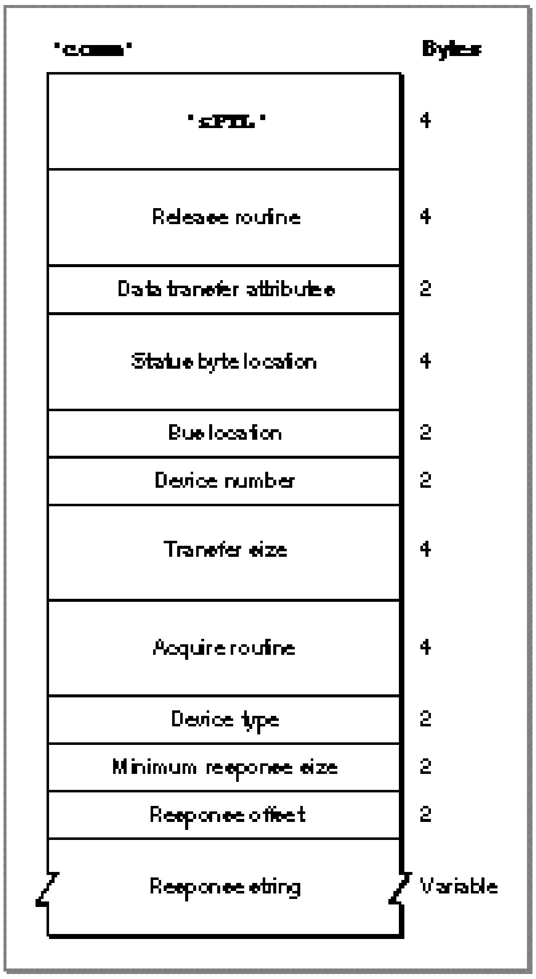
Listing 6-12 An example of a PrinterShare communications resource

```
resource gxDeviceCommunicationsType (-4094, sysHeap, purgeable)
{
   PrinterShare,
      {
      "",            /* AppleTalk address; Chooser fills in */
      0
      };
};
```

SCSI Communications Resource

You need to define a SCSI communications resource for SCSI output devices. The structure of this form of a communications resource is shown in Figure 6-15. Although QuickDraw GX provides the SCSI communications resource, it does not provide a default implementation of the device communications messages for SCSI devices. If you are implementing a printer driver for a SCSI device, you must override the device communications messages, which are described in the chapter “Printing Messages” in this book.

Figure 6-15 The SCSI communications resource



The SCSI communications resource consists of a communications type identifier and a number of SCSI control values.

- Type identifier. 'sPTL'(or the constant SCSI) is used to identify SCSI communications resources.
- Release routine. An internally used value. This must be 0 in your resource definition.
- Data transfer attributes. The SCSI input/output attributes that apply to data transfers.
- Status byte location. An internally used value. This must be 0 in your resource definition.
- Bus location. The SCSI bus number for the device. The value 0 indicates the motherboard.
- Device number. The SCSI device number for the device.
- Transfer size. The number of bytes for each transfer (at the SCSI TIB level). If this value is 0, it is ignored.
- Acquire routine. An internally used value. This must be 0 in your resource definition.
- Device type. The type of SCSI device to look for.
- Minimum response size. The minimum amount of data, in bytes, in the response data.
- Response offset. The location in the response data to look for the response string. This is specified as a byte offset from the start of the response data.
- Response string. The string to look for in the response data.

Listing 6-13 shows an example of a SCSI communications resource that defines a SCSI communications connection for a LaserWriter IISC printer.

Listing 6-13 An example of a SCSI communications resource

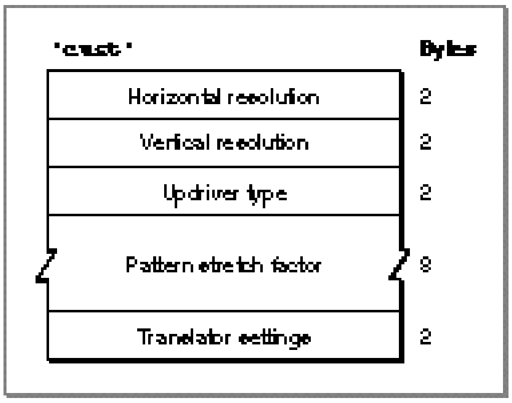
```
resource gxDeviceCommunicationsType (-4095, sysHeap, purgeable)
{
   SCSI
   {
      0,          /* release device routine */
      0,          /* I/O attibutes */
      0,          /* status byte */
      0,          /* SCSI bus number */
      0,          /* SCSI ID */
      0,          /* chunk level */
      0,          /* acquire routine (use default) */
      2,          /* device type to look for */
      27,         /* minimum length of additional response data */
      8,          /* bytes from start for search */
      "APPLE   PERSONAL LASER  ",   /* search string */
   };
};
```

Note
QuickDraw GX provides the SCSI communications resource type and supports it in the Chooser. However, QuickDraw GX does not provide a default implementation of SCSI printer communications. If you are implementing a driver for a SCSI printer, you must override all of the communications messages. These messages are described in the chapter "Printing Messages" in this book. u

The Customization ('cust') Resource

The driver customization (`'cust'`) resource, of type `gxCustType`, tells QuickDraw GX how to support the use of your printer driver by Macintosh Printing Manager applications. It defines values that help QuickDraw GX to translate Macintosh Printing Manager calls into messages for your printer driver. If you do not include a customization resource for your driver, a default version is used by QuickDraw GX. The default version works well for most drivers. Figure 6-16 shows the structure of a customization resource.

Figure 6-16 The customization resource



The customization resource defines the following:

- Horizontal resolution. The horizontal resolution to use for the device in dots per inch (dpi).
- Vertical resolution. The vertical resolution to use for the device in dots per inch.
- Macintosh Printing Manager interface type. The style of Macintosh Printing Manager interface driver with which your driver is compatible. Constants for this field are shown in Table 6-20.
- Pattern stretch factor. A point that indicates how to scale bitmap patterns when printing.

- Translator settings. Settings for translating Macintosh Printing Manager driver calls into messages for your driver. Constants for the translator setting values, which you can combine together into a single value, are shown in Table 6-21.

Table 6-20 shows the constants that you can use to specify the updriver type in customization resources.

Table 6-20 Updriver values for the customization resource

Constant	Value	Explanation
`defaultUpDriver`	0	LaserWriter interface
`laserWriter`	0	LaserWriter interface
`laserWriterSC`	1	LaserWriter SC interface

Table 6-21 shows the constants that you can use to specify translator settings in customization resources.

Table 6-21 Translator setting values

Constant	Value	Explanation
`gxDefaultOptionsTranslation`	0x0000	These default settings produce the most accurate representation of the QuickDraw data.
`gxOptimizedTranslation`	0x0001	The translator fills in the center of outline text shapes that are drawn in `srcOr` mode with white, producing the same effect as black-and-white QuickDraw.
`gxReplaceLineWidthTranslation`	0x0002	The line width specified in the PostScript `SetLineWidth` command becomes the new line width. The default (without this option) is to scale the line width by the specified value.
`gxSimpleScalingTranslation`	0x0004	The translator scales data via a simple multiply, without compensating for increased resolution. The resulting scaled image is similar to what QuickDraw would have produced when it scaled the data.
`gxSimpleGeometryTranslation`	0x0008	The translator translates QuickDraw data without taking into account the QuickDraw hanging pen. The translator does not reproduce the QuickDraw 6-sided line and 7-sided triangle. This means faster drawing with QuickDraw GX with some loss of accuracy. This option turns on the `gxSimpleScalingTranslation` option.

Table 6-21 Translator setting values (continued)

Constant	Value	Explanation
`gxSimpleLinesTranslation`	0x000C	The translator maintains the width of lines when lines are at an angle, rather than producing a thicker diagonal line. With this option, line width is the same as pen width. This option turns on the `simpleScalingTranslation` and `simpleGeometryTranslation` options.
`gxLayoutTextTranslation`	0x0010	The translator allows layout to perform default substitutions, which produce text that is more attractive than, but different from, the original QuickDraw text.
`gxRasterTargetTranslation`	0x0020	The translator produces output for a raster device.
`gxPostScriptTargetTranslation`	0x0040	The translator produces output for a PostScript device.

The customization resource ID needs to be the constant `gxCustID` (-8192). Listing 6-14 shows an example of a customization resource for the ImageWriter II printer driver, which uses the optimized translation settings and scales bitmaps by a factor of 2 both horizontally and vertically.

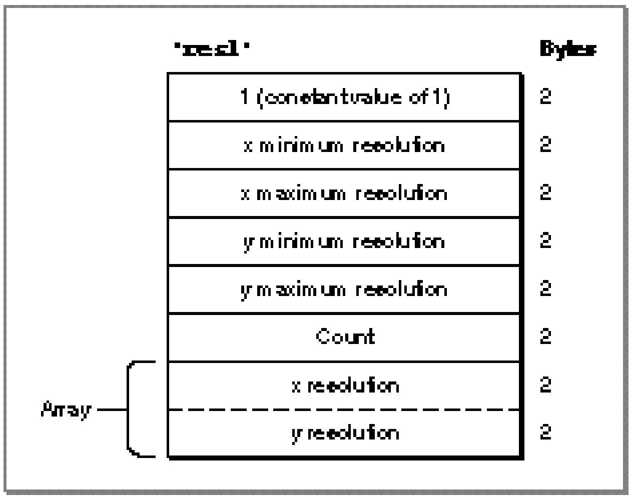
Listing 6-14 An example of a customization resource

```
resource gxCustType (gxCustID, sysHeap, purgeable)
{
   144, 144,      /* 144 dpi device */
   defaultUpDriver,     /* use default upDriver */
   {2,2},               /* pattern scaling (144dpi/72dpi)*/
   gxOptimizedTranslation/* use optimized translator settings */
};
```

The Resolution ('resl') Resource

The resolution (`'resl'`) resource, of type `gxReslType`, is used to define the resolution values that a driver supports. This resource is optional: if you do not provide one, the resolution values in the customization resource are used. If you do not provide a resolution resource or a customization resource, the default customization values (the defaults for the updriver) are used. You can define a range of resolution values in a single resolution resource. Figure 6-17 shows the structure of resolution resource.

Figure 6-17 The resolution resource



The resolution resource defines the resolution range for a driver and contains a variable number of specific resolution entries.

- Range type. A constant value of 1.
- x minimum resolution. The minimum horizontal resolution supported by the driver in dots per inch (dpi).
- x maximum resolution. The maximum horizontal resolution supported by the driver in dpi.
- y minimum resolution. The minimum vertical resolution supported by the driver in dpi.
- y maximum resolution. The maximum vertical resolution supported by the driver in dpi.
- Count. The number of resolution pairs that follow in this resource.

Two values are defined for each specific resolution entry.

- x resolution. The horizontal resolution of this entry in dpi.
- y resolution. The vertical resolution of this entry in dpi.

The resolution resource ID needs to be –8192. Listing 6-15 shows an example of a resolution resource.

Listing 6-15 An example of a resolution resource

```
resource gxReslType (-8192, sysHeap, purgeable)
{
   rangeType,
   25, 1500,
   25, 1500,
   {
   300, 300;
   600, 600;
   };
};
```

This resource is for a printing device that supports resolutions from 25 to 1500 dpi. The recommended resolutions values are 300 and 600 dpi.

The Print Record ('PREC') Resource

The print record (`'PREC'`) resource defines a default print record for a driver, which is used to create `TPrint` records for compatibility with Macintosh Printing Manager applications. When the user of a Macintosh Printing Manager application prints a document, this resource is used to define the initial values to use for printing that document. This is an optional resource. If you don't supply one, QuickDraw GX uses the Apple default values.

The Apple default version of the `'PREC'` resource is 120 bytes long and matches the structure of the universal print structure, which is described in the section "The Universal Print Structure" beginning on page 4-12 in the chapter "Printing Messages."

The structure of the print record resource is driver dependent. Many printer drivers use the Apple default values and then modify the universal print structure fields to meet their needs.

Listing 6-16 shows an example of a print record resource that is the print record for the ImageWriter II printer driver.

Listing 6-16 An example of a print record resource

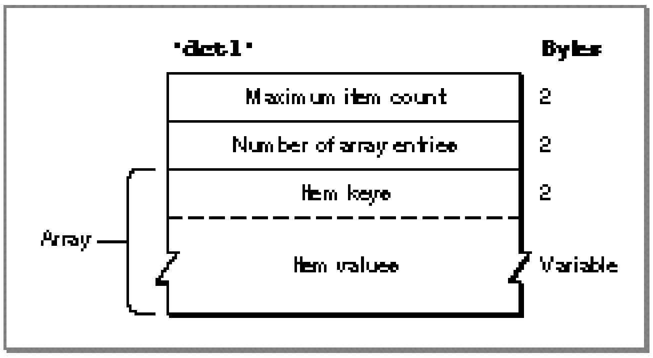
```
resource 'PREC' (0, sysHeap, purgeable) {
   4,               /* increase from 3 to 4 for QuickDraw GX */
   0,
   72,
   80,
   {0, 0, 752, 640},
   {-36, -20, 756, 660},
   1,
   0,
   NoDftBts,
   NoResSet,
   NoScroll,
   NoZoom,
   NormPix,
   Portrait,
   LowRes,
   1320,
   1020,
   0,
   Fanfold,
   0,
   72,
   80,
   {0, 0, 752, 640},
   80,
   32,
   640,
   3200,
   24,
   norm,
   1,
   1,
   1,
   scanTB,
   1,
   9999,
   1,
   Spool,
   1,
   0,
   0,
   0,
   0,
   0,
   0,
   1
};
```

The Dialog Control ('dctl') Resource

You need to include a dialog control resource to define control items for the print dialog boxes used with Macintosh Printing Manager applications. This resource provides you with a means of mapping the behavior of items in an item list (`'DITL'`) resource to fields in the universal print structure that QuickDraw GX uses for compatibility with Macintosh Printing Manager applications. The resource consists of a list of action types, each of which maps to one or more items in the item list resource and to a location in the universal print record.

The item list resource is described in *Inside Macintosh: Macintosh Toolbox Essentials*. The universal print structure is described in the section “The Universal Print Structure” beginning on page 4-12 in the chapter “Printing Messages.” Figure 6-18 shows the structure of the dialog control resource.

Figure 6-18 The dialog control resource



The dialog control resource contains a count of the control-item entries and an array of control-item definitions.

- Maximum item count. The maximum item count for the item list resource.
- Number of array entries. The number of control-item definitions that follow.

Each control item definition has a unique item-key value and some number of values.

- Item key. The type of control item. The values that you can use in this field are shown in Table 6-22.
- Item values. The values that are needed to define the control item. The number and types of the values that you include in this field depend on the type of item that you are defining, as shown in the individual action sections beginning on page page 6-55.

The types of control items that you can define in a dialog control resource are shown in Table 6-22. Each item is described in more detail in the sections that follow the table.

Table 6-22 Control item types

Item type	Value	Explanation
`Button`	1	Implements the Cancel button
`Cluster`	2	Implements options that are presented to the user as a series of radio buttons
`Copies`	3	Implements the editable text item into which the user enters the number of copies to be printed

continued

Table 6-22 Control item types (continued)

Item type	Value	Explanation
`DialogBtn`	4	Implements dialog boxes on top of the current dialog box
`Frill`	5	Used to draw the double-line bar, the version number, the heavy outline around the default button, and the printer name in the title of the dialog box
`Moof`	6	Draws Moof, the circus trick dog, which is displayed in the Page Setup dialog box to denote page orientation
`OKButton`	7	Implements the OK button
`Orientation`	8	Implements the orientation icon buttons
`PageRange`	9	Allows the user to specify the range of pages to be printed
`PaperSizes`	10	Implements selection of paper sizes by the user
`Scale`	11	Implements the print-scaling (Reduce / Enlarge) option
`Toggle`	12	Implements checkboxes that set and clear various bits in the `iFlags` field of the universal print structure

A dialog control resource can contain a number of actions, each of which is named by its type and followed by a number of values. Listing 6-17 shows two definitions of a dialog control resource from the ImageWriter II printer driver.

Listing 6-17 Two examples of a dialog control resource

```
resource 'dctl' (-8192, sysHeap, purgeable) {
   20,
   {
   Button { 2, cancel },
   Frill { 4, line },
   PaperSizes { 0, 0, { 6, 7, 8, 9, 10, 11 } },
   Orientation { 13, 14, 0, 0 },

   Toggle { 16, bPreciseBitmap },
   Toggle { 17, bUser0 },
   Toggle { 18, bBiggerPages },

   Frill { 19, version },
   Frill { 20, default },
   }
};
```

```
resource 'dctl' (-8191, sysHeap, purgeable) {
   21,
   {
   Button { 2, cancel },
   Frill { 4, line },
   Cluster { quality, { 6, 7, 8 } },
   PageRange { 10, 11, 12, 14},
   Copies { 16 },
   Cluster { feed, { 18, 19 } },

   Frill { 20, version },
   Frill { 21, default },
   }
};
```

The following sections describe the values that you need to specify in the dialog control resource for each item type.

Button Actions

You use the button (`Button`) action-item type to implement the Cancel button in the Print dialog box used with Macintosh Printing Manager applications. When the user selects this button, QuickDraw GX automatically restores the values in the universal print structure to what they were when the dialog box was created. This action uses two values:

- Item ID. The integer ID of the item in the item list to which this button pertains.
- Button kind. The kind of button that this represents. At this time, you can only use this item type to define Cancel buttons. You use the integer constant `cancel` to do this.

Listing 6-17 on page 6-54 contains two examples of button actions.

Cluster Actions

You use the cluster (`Cluster`) action-item type to implement clusters of radio buttons. When the user selects one of the radio buttons, QuickDraw GX automatically updates the corresponding value in the universal print structure. This action uses two values:

- Cluster type. The kind of item that the cluster is used to define. This is one of the integer constants shown in Table 6-23.
- Cluster items. The IDs of the items to which the cluster corresponds.

Table 6-23 shows the constants that you can use to define what kind of item the radio button cluster is defining. Listing 6-17 contains an example of a cluster action.

Table 6-23 Cluster type constants for the dialog control resource

Cluster type	Value	Explanation
`feed`	0	Defines the feed mode used for the print job by filling in the `feed` field in the universal print structure
`quality`	1	Defines the print-quality mode used by filling in the `qualityMode` field in the universal print structure
`coverPage`	2	Defines the type of cover page to print for the print job by filling in the `coverPage` field in the universal print structure
`firstPage`	3	Defines which of the printer's paper trays is used as the first paper tray for this print job by filling in the `firstTray` field in the universal print structure
`restPage`	4	Defines the other paper tray for this print job by filling in the `remainingTray` field in the universal print structure
`headMotion`	5	Defines the type of print-head motion to use for the print job by filling in the `headMotion` field in the universal print structure
`createFile`	6	Defines the type of file to save for this print job by filling in the `saveFile` field in the universal print structure
`user0`	7	Reserved to allow the driver to define its own clusters and fill in the `userCluster1` field in the universal print structure
`user1`	8	Reserved to allow the driver to define its own clusters and fill in the `userCluster2` field in the universal print structure
`user2`	9	Reserved to allow the driver to define its own clusters and fill in the `userCluster3` field in the universal print structure

Copies Actions

You use the copies (`Copies`) action-item type to implement the editable text item into which the user enters the desired number of print copies. When the user types a value into this field, QuickDraw GX verifies that the number is between 1 and 9999 and then stores that value in the `actualCopies` field in the universal print structure. This action uses one value:

- Item ID. The ID of the item to which this action pertains.

Listing 6-17 on page 6-54 contains an example of the copies action type.

Dialog-Button Actions

You use the dialog-button (`DialogBtn`) action-item type to implement dialog boxes that appear on top of the current dialog box. For example, in the LaserWriter printer driver, the Options button in the style dialog box displays the dialog box with Moof in it on top of the style dialog box. When the user selects the item, QuickDraw GX brings up the specified dialog box, which can itself contain any of the action-item types in it. This action uses three values:

- Item ID. The ID of the item to which this action pertains.
- Dialog ID. The ID of the dialog resource to display when this item is selected by the user.
- Control ID. The ID of the dialog control resource for the dialog box that is displayed when this item is selected by the user.

Frill Actions

You use the frill (`Frill`) action-item type to implement informational and decorative objects in the dialog box. This action uses two values:

- Item ID. The ID of the item to which this action pertains.
- Frill type. The type of frill. This is one of the integer values shown in Table 6-24.

Table 6-24 Frill type constants for the dialog control resource

Frill type	Value	Explanation
`line`	0	Used to draw the double-line bar in the dialog box.
`version`	1	Used to draw the version number in the dialog box.
`default`	2	Used to draw the heavy outline under the default button in the dialog box.
`printerName`	3	Used to draw the printer name in the dialog box. This item must be a static text item.

Listing 6-17 on page 6-54 contains several examples of frill actions.

Moof Actions

You use the Moof (`Moof`) action-item type to display Moof, the circus trick dog, as is done when the orientation of the page is displayed in the Page Setup dialog box. This action uses one value:

- Item ID. The ID of the item in which Moof is drawn when the dialog box is initialized.

Confirm-Button Actions

You use the confirm-button (`OkButton`) action-item type to implement the OK button in a dialog box. This item is only needed if there is a Create File cluster that needs to change the name of the OK button for printing (to switch the name of the OK button depending on whether the user is printing or saving a file). This action uses three values:

- Item ID. The ID of the OK button item. This value must be 1.
- String ID 1. The ID of the string resource to display for the “Print” choice.
- String ID 2. The ID of the string resource to display for the “Save” choice.

Orientation Actions

You use the orientation (`Orientation`) action-item type to implement the orientation icon buttons. You can use up to four different buttons to establish the value of the `orientation` field in the universal print structure. QuickDraw GX assumes that the driver contains as many icon (`'ICON'`) resources as there are nonzero entries in this item definition. QuickDraw GX also assumes that the icon resource IDs start at 0 and increase by 1. This action uses four values:

- Portrait item ID. The ID of the portrait orientation item.
- Landscape item ID. The ID of the landscape orientation item.
- Alternate portrait item ID. The ID of the rotated portrait orientation item.
- Alternate landscape item ID. The ID of the rotated landscape orientation item.

Page-Range Actions

You use the page-range (`PageRange`) action-item type to implement the page-range selection. When the user enters page values in the “From” and “To” fields, QuickDraw GX updates the `firstpage` and `lastPage` values in the universal print structure. This action uses four values:

- All pages button ID. The ID of the “All Pages” button.
- Range button ID. The ID of the “Range” button.
- From field ID. The ID of the “From” or first-page editable text field.
- To field ID. The ID of the “To” or last-page editable text field.

Listing 6-17 on page 6-54 contains an example of a page-range action in a dialog control resource.

Paper-Size Actions

You use the paper-size (`PaperSizes`) action-item type to implement selection of paper sizes for a print job. This action has two values and an array of item IDs:

- Pop-up item ID. The item ID of the paper-sizes pop-up menu.
- Pop-up radio item ID. The ID of the paper-sizes pop-up menu.

- Item IDs. An array of IDs, one per radio button item that is referenced by this action.

Listing 6-17 on page 6-54 contains an example of a paper-size action item in a dialog control resource.

Scale Actions

You use the scale (`Scale`) action-item type to implement the reduce and enlarge option in the Page Setup dialog box. This action uses two values:

- Edit ID. The ID of the editable text item that contains the scaling value.
- Arrow ID. The ID of the item that controls the up and down actions on the scaling value.
- Scaling table ID. The resource ID of the scaling table resource that defines the scaling values for this action item. The scaling table resource is described on page 6-61.

Toggle Actions

You use the toggle (`Toggle`) action-item type to implement the checkboxes that set and clear the flags in the `options` field of the universal print structure. This action uses two values:

- Item ID. The ID of the item to which this action pertains.
- Toggle type. The kind of toggle item to which this action pertains. You can combine any of the integer constants from Table 6-25 to form the value of this field. When you combine values and the user clicks the checkbox, all of the flags associated with the value are set.

Table 6-25 Toggle action-item flags for the dialog control resource

Toggle item type	Value	Explanation
`bPreciseBitmap`	0x0001	The driver needs to format pages as tall-adjusted for the Apple ImageWriter family of printers, and the driver needs to use precise bitmaps for the Apple LaserWriter family of printers. This is the same as the `gxPreciseBitmap` value described on page 4-15 in the chapter "Printing Messages."
`bBiggerPages`	0x0002	The driver needs to not apply gaps if printing this job to one of the Apple ImageWriter family of printers, and the driver needs to use a large print area for the Apple LaserWriter family of printers. This is the same as the `gxBiggerPages` value described on page 4-15 in the chapter "Printing Messages."

continued

Table 6-25 Toggle action-item flags for the dialog control resource (continued)

Toggle item type	Value	Explanation
`bGraphicSmoothing`	0x0004	The driver needs to perform graphics smoothing on the Apple LaserWriter family of printers. This is the same as the `gxGraphicSmoothing` value described on page 4-15 in the chapter "Printing Messages."
`bTextSmoothing`	0x0008	The driver needs to perform text smoothing for this print job. This is the same as the `gxTextSmoothing` value described on on page 4-15 in the chapter "Printing Messages."
`bFontSubstitution`	0x0010	The driver needs to perform font substitution for this print job. This is the same as the `gxFontSubstitution` value described on page 4-15 in the chapter "Printing Messages."
`bInvert`	0x0020	The driver needs to invert the printed image (convert white to black and black to white) for this print job. This is the same as the `gxInvertPage` value described on page 4-15 in the chapter "Printing Messages."
`bFlipHoriz`	0x0040	The driver needs to flip pages horizontally for this print job. This is the same as the `gxFlipPageHoriz` value described on page 4-15 in the chapter "Printing Messages."
`bFlipVert`	0x0080	The driver needs to flip pages vertically for this print job. This is the same as the `gxFlipPageVert` value described on page 4-15 in the chapter "Printing Messages."
`bColorMode`	0x0100	This print job uses color printing. This is the same as the `gxColorMode` value described on page 4-15 in the chapter "Printing Messages."
`bBidirectional`	0x0200	This print job uses bidirectional printing. This is the same as the `gxBidirectional` value described on page 4-15 in the chapter "Printing Messages."
`bUser0`	0x0400	Available for driver-defined options.
`bUser1`	0x0800	Available for driver-defined options.
`bUser2`	0x1000	Available for driver-defined options.
`bReserved0`	0x2000	Reserved for future options.
`bReserved1`	0x4000	Reserved for future options.
`bReserved2`	0x8000	Reserved for future options.

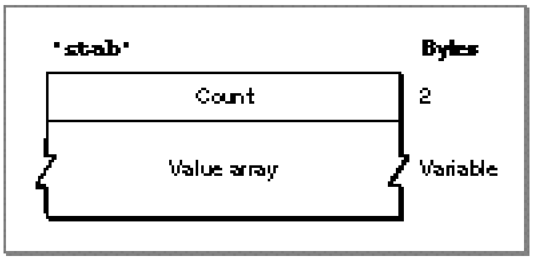
Listing 6-17 on page 6-54 contains several examples of toggle action items in a dialog control resource.

The Scaling Table ('stab') Resource

You need to include a scaling table resource when you include a scale action-item type in your dialog control (`'dctl'`) resource, which is described in the previous section. The values in the scaling table resource define the discrete page-size reduction values that a user can select for a Macintosh Printing Manager application.

Figure 6-19 shows the structure of the scaling table resource.

Figure 6-19 The scaling table resource



The scaling table resource contains a count of the scaling value entries and an array of values.

- Count. The number of scaling values that follow.
- Value array. An array of scaling values. Each of these integers defines a discrete scaling value that the user can select when clicking on the up and down arrows for the scaling item in the dialog box.

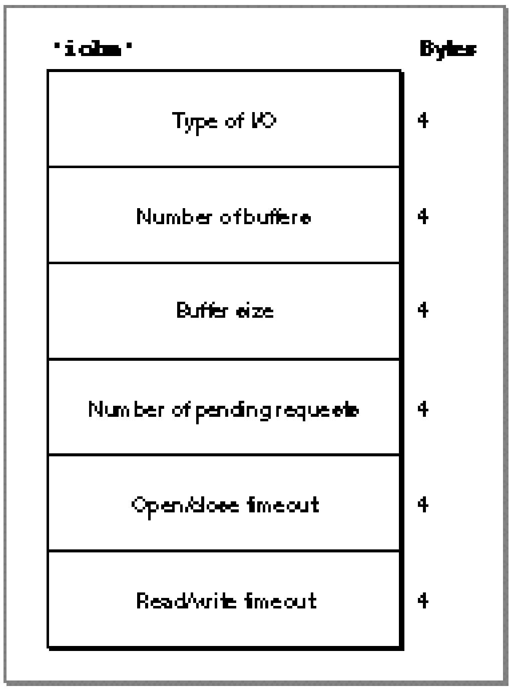
Note
The user can directly enter any scaling value into the scaling item box. The values in the scaling table are only used when the user clicks on the up and down arrows. u

IMPORTANT
The values in the scaling table resource must be in ascending order. s

The Buffering and Input/Output Preferences ('iobm') Resource

The buffering and input/output preferences (`'iobm'`) resource, of type `gxUniveralIOPrefsType`, defines the timeout and buffering parameters for the driver. This resource is optional; if you do not supply one, QuickDraw GX uses standard I/O with two buffers, each 1 KB long, and defines each timeout value to be 10 seconds. Figure 6-20 shows the structure of the buffering and input/output preferences resource.

Figure 6-20 The buffering and input/output preferences resource



The buffering and input/output preferences resource defines the buffering and timeout values for I/O.

- Type of I/O. The type of I/O to use. This is one of two values: `standardIO` or `customIO`. If you specify `customIO`, it means that your driver is handling input and output communications without using the built-in support.

Note
SCSI drivers must use the `customIO` value in this field. QuickDraw GX does not provide a default implementation of SCSI connections. If you do develop a SCSI driver, you must override the device communications messages and manage all of the communications yourself. u

- Number of buffers. The number of buffers to create for this driver.
- Buffer size. The size of each buffer.
- Number of pending requests. The maximum number of input or output requests that can be pending at any time.
- Open/close timeout. The number of clock ticks that constitute a timeout when trying to open or close the device.
- Read/write timeout. The number of clock ticks that constitute a timeout when trying to read from or write to the device.

The buffering and input/output preferences resource ID needs to be the constant `gxUniversalIOPrefsID`. Listing 6-18 shows an example of a buffering and input/output preferences resource.

Listing 6-18 An example of a buffering and input/output preferences resource

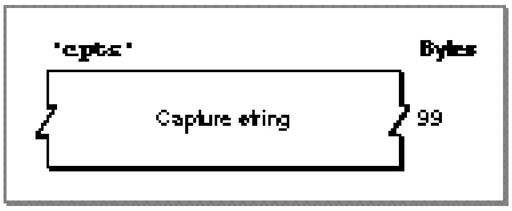
```
resource gxUniveralIOPrefsType (gxUniversalIOPrefsID, sysHeap,
                                                     purgeable)
{
   standardIO,
   4,             /* 4 buffers */
   2048,          /* each buffer 2 KB */
   10,            /* up to 10 I/O operations pending */
   1200,          /* open/close timeout of 1200 clock ticks */
   1200           /* read/write timeout of 1200 clock ticks */
   };
};
```

This resource is for the ImageWriter II printer driver, which uses four buffers, each 2 KB long. Ten request blocks will be allocated for communications with this device, and requests to open, close, read, or write the device will time out in 1200 clock ticks (20 seconds).

The Capture ('cpts') Resource

The capture (`'cpts'`) resource, of type `gxCaptureType`, allows devices to be removed and replaced on a network during printer sharing (through the `GXCaptureOutputDevice` message). This resource is not needed if your driver does not support the capture-and-release concept. The default version of the `GXCaptureOutputDevice` messages uses these resources and works only for PAP devices. Figure 6-21 shows the structure of a capture resource.

Figure 6-21 The capture resource



- Capture string. The string that specifies capture information. There are four kinds of capture information, each of which you specify in a separate capture resource. The four capture-string resource type constants are shown in Table 6-26.

Table 6-26 Capture resource types

Constant	Explanation
`gxCaptureStringID`	Used to capture and release the device and does not contain a length byte
`gxReleaseStringID`	Used for a released device and does not contain a length byte
`gxUncapturedAppleTalkType`	The AppleTalk device name that is used for an uncaptured device; it includes a length byte
`gxCapturedAppleTalkType`	The AppleTalk device name that is used for a captured device; it includes a length byte

The system performs string substitution in any of the capture strings and defines constants that you can use for this purpose. Each of these constants is replaced with the appropriate value at run time. The constant strings are shown in Table 6-27.

Table 6-27 Substitution strings for the capture resource

Constant	Explanation
`PRINTERNAME`	Replaced with the name of the printer
`PRINTERTYPE`	Replaced with the name-binding protocol (NBP) type of the printer
`NAMELEN`	Replaced with a single byte that is the length of the printer name
`TYPELEN`	Replaced with a single byte that is the length of the printer NBP type

The four capture resources shown in Listing 6-19 are taken from the ImageWriter II printer driver.

Listing 6-19 Examples of capture resources for a printer driver

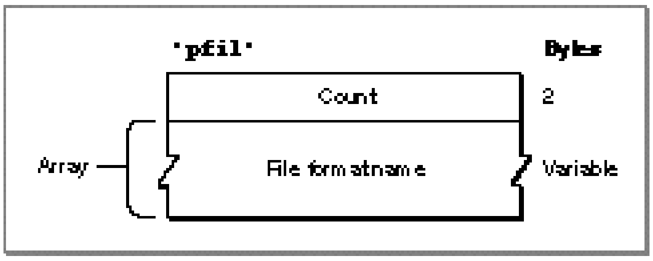
```
resource gxCaptureType (gxCapturedAppleTalkType, sysHeap,
                                                purgeable)
{
   "\0D011ImageShared"
};

resource gxCaptureType (gxUncapturedAppleTalkType, sysHeap,
                                                  purgeable)
{
   "\0D011ImageWriter"
};

resource gxCaptureType (gxCaptureStringID, sysHeap, purgeable)
{
   "\0X1B\0X62NAMELENPRINTERNAMETYPELENPRINTERTYPE\0X01*"
};

resource gxCaptureType (gxReleaseStringID, sysHeap, purgeable)
{
   "\0X1B\0X62NAMELENPRINTERNAMETYPELENPRINTERTYPE\0X01*"
};
```

The Print-File Formats ('pfil') Resource

The print-file formats resource is used to specify driver-specific output file formats to display to the user for the print-to-file option. Each format name specified in this resource is displayed in the list of file formats shown in the standard file dialog box when the user confirms the Print dialog box after selecting the “Destination” option.

Figure 6-22 shows the structure of a print-file formats resource.

Figure 6-22 The print file formats resource



The print-file formats resource consists of a count and a number of file format names.

- Count. The number of file format names that follow in this resource.
- File format name. The Pascal string name to use when creating the output file.

Listing 6-20 shows an example of a print-file formats resource that defines the file format name for the LaserWriter IIg printer driver.

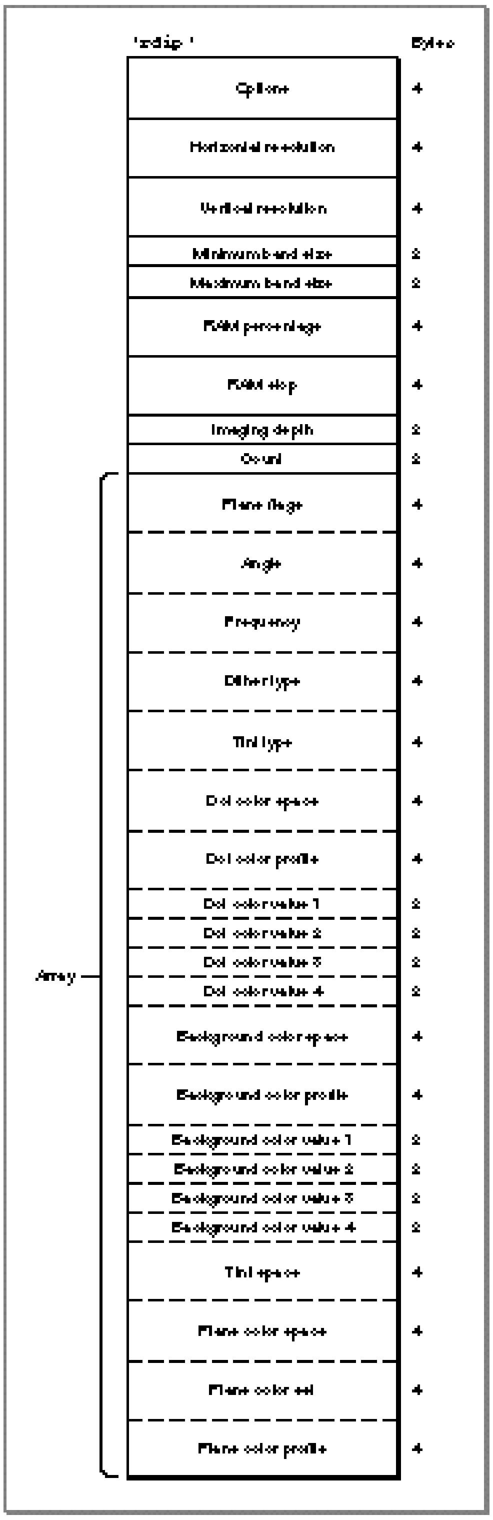
Listing 6-20 An example of a print-file formats resource

```
resource 'pfil' ( gxDriverFileFormatID, sysHeap, purgeable) {

   {
      "PostScript™"
   };
};
```

The Raster Preferences ('rdip') Resource

The raster preferences (`'rdip'`) resource, of type `gxRasterPrefsType`, controls the rendering options for a raster driver. This resource is required for all raster printer drivers. Figure 6-23 shows the structure of a raster preferences resource.

Figure 6-23 The raster preferences resource



The raster preferences resource contains a number of values that define how imaging is performed on the device, followed by an array of plane values, each of which defines parameters for a specific color on the printer. For example, if your driver accepts CMYK (cyan, magenta, yellow, and black color specification) data, QuickDraw GX sends it planes for cyan, magenta, yellow, and black. The values in the plane array correspond to the number of times that you need to send raster data to the printer for color printing.

You need to understand how QuickDraw GX represents and uses colors to understand many of the values in this resource. Refer to two chapters in *Inside Macintosh: QuickDraw GX Objects:* the "View Objects" chapter and the "Colors and Color-Related Objects" chapter.

- Options. A collection of option values that you can combine together to specify how rendering occurs on a raster device. Constants for the raster preference options are shown in Table 6-28 on page 6-70.
- Horizontal resolution. The horizontal resolution for imaging as a fixed point number.
- Vertical resolution. The vertical resolution for imaging as a fixed point number.
- Minimum band size. The minimum band size, in pixels.
- Maximum band size. The maximum band size, in pixels. A value of 0 indicates that the maximum band size is the whole page.
- RAM percentage. The amount of available temporary memory that can be used. The actual amount used is computed by the following calculation:

  ```
  ((Free memory - RAMSlop) * RamPercentage);
  ```

  The result is then rounded into a multiple of the minimum band size, but is never made larger than the maximum band size.
- RAM slop. The amount to subtract from the total available temporary memory before computing the percentage, as shown above.
- Imaging depth. The number of pixels per plane for imaging.
- Count. The number of entries in the plane array for this resource. This value is filled in at run time and is not part of the resource definition.

Each entry in the plane array is for a specific color. For example, a driver that accepts CMYK data is sent planes for cyan, magenta, yellow, and black. This data may or may not be equivalent to a band on a ribbon, an ink color, or a wax color; however, each plane that you define in this array does represent one pass of data by the device. Each entry in the plane array contains the following data values.

- Plane flags. A collection of flag values that you can combine together to specify controls for rendering the plane. Constants for the plane flags are shown in Table 6-29 on page 6-70.
- Angle. The apparent direction to use for halftones, in degrees. This is a fixed point value. The chapter "View Objects" in *Inside Macintosh: QuickDraw GX Objects* describes using halftones.
- Frequency. The size of the dot, in cells per inch, to use for halftones. This is a fixed point value. The chapter "View Objects" in *Inside Macintosh: QuickDraw GX Objects* describes using halftones.

- Dither type. The dithering style, or halftone method, to use for this plane. Constants for color-plane dithering types are shown inTable 6-30 on page 6-71. The results of using these values are shown in the discussion of halftones in the chapter “View Objects” in *Inside Macintosh: QuickDraw GX Objects.*
- Tint type. The type of tinting to apply to this plane. Constants for the plane tinting types are shown in Table 6-31 on page 6-71. The use of these values is explained in the chapter “View Objects” in *Inside Macintosh: QuickDraw GX Objects.*
- Dot color space. Which color space the values in this plane use. The constants for color-space values are shown in the chapter “Colors and Color-Related Objects” in *Inside Macintosh: QuickDraw GX Objects.*
- Dot color profile. The color profile to use for the dot color on this plane. The value `gxNoProfile` (0) indicates that a color profile is not to be used. Any other value specifies the ID of the color matching (`'cmat'`) resource to use. Color profiles are explained in the chapter “Colors and Color-Related Objects” in *Inside Macintosh: QuickDraw GX Objects.*

  If you specify `gxNoProfile,` no color matching is performed.
- Dot color value 1. The first value for defining the dot color for this plane.
- Dot color value 2. The second value for defining the dot color for this plane.
- Dot color value 3. The third value for defining the dot color for this plane.
- Dot color value 4. The fourth value for defining the dot color for this plane.
- Background color space. The color space to use for the background color on this plane. The color-space values are shown in chapter “Colors and Color-Related Objects” in *Inside Macintosh: QuickDraw GX Objects.*
- Background color profile. The color profile to use for the dot color on this plane. The value `gxNoProfile` (0) indicates that a color profile is not to be used. Any other value specifies the ID of the color matching (`'cmat'`) resource to use. Color profiles are explained in the chapter “Colors and Color-Related Objects” in *Inside Macintosh: QuickDraw GX Objects.*

  If you specify `gxNoProfile,` no color matching is performed.
- Background color value 1. The first value for defining the background color for this plane.
- Background color value 2. The second value for defining the background color for this plane.
- Background color value 3. The third value for defining the background color for this plane.
- Background color value 4. The fourth value for defining the background color for this plane.
- Tint space. The color space to use for halftoning. Use one of the values shown in the chapter “Colors and Color-Related Objects” in *Inside Macintosh: QuickDraw GX Objects.*
- Plane color space. The color space of the plane (the original data’s color space). Color spaces, color sets, and color profiles are described in the chapter “Colors and Color-Related Objects” in *Inside Macintosh: QuickDraw GX Objects.*

- Plane color set. The resource ID of the color set to use for the plane. You can specify the value `gxNoSet` to indicate no color set.
- Plane color profile. The resource ID of the color profile to use for the plane. You can specify the value `gxNoProfile` to indicate no color profile.

Table 6-28 shows the constants that you can use to specify the option values for the raster preferences resource.

Table 6-28 Raster preference option values

Constant	Value	Explanation
`gxDefaultRaster`	0	Uses default options
`gxDontResolveTransferModes`	0x01	Tells the system to not resolve transfer modes because your 32-bit device can do it faster on its own
`gxRenderInReverse`	0x02	Traverses the image in reverse order
`gxOnePlaneAtATime`	0x04	Renders each plane separately
`gxSendAllBands`	0x08	Sends bands, even if empty (all white)

Table 6-29 shows the constants that you can use to specify the plane flags values for the raster preferences resource.

Table 6-29 Flags used for each plane in the raster preferences resource

Constant	Value	Explanation
`gxDefaultOffscreen`	0	Specifies that the default settings are used, which means that the view port's halftone is based on the information in the imaging system data.
`gxDontSetHalftone`	1	Specifies that the system must not set the view port's halftone. Use this option when you do not want halftones on a device.
`gxDotTypeIsDitherLevel`	2	Specifies that the system must set the view port's dither level to the value stored in the `gxDotType` parameter and ignore other halftone information.

Table 6-30 shows the constants that you can use to specify the plane dithering types for the raster preferences resource.

Table 6-30 Plane dithering types

Constant	Value
`gxRoundDot`	1
`gxSpiralDot`	2
`gxSquareDot`	3
`gxLineDot`	4
`gxEllipticDot`	5
`gxTriangleDot`	6
`gxDispersedDot`	7

Table 6-31 shows the constants that you can use to specify the plane tinting types for the raster preferences resource.

Table 6-31 Plane tinting types

Constant	Value	Explanation
`gxLuminanceTint`	1	Use the luminance of the color
`gxAverageTint`	2	Add all the components, then divide by the number of components
`gxMixtureTint`	3	Find the closest color on the axis between the foreground and background colors
`gxComponent1Tint`	4	Use the value of the first component of the color space
`gxComponent2Tint`	5	Use the value of the second component of the color space
`gxComponent3Tint`	6	Use the value of the third component of the color space
`gxComponent4Tint`	7	Use the value of the fourth component of the color space

The ID of the raster preferences resource must be the constant `gxRasterPrefsID`. Listing 6-21 shows an example of a raster preferences resource for a 144-dpi four-plane color device, the ImageWriter II raster printer.

Listing 6-21 An example of a raster preferences resource

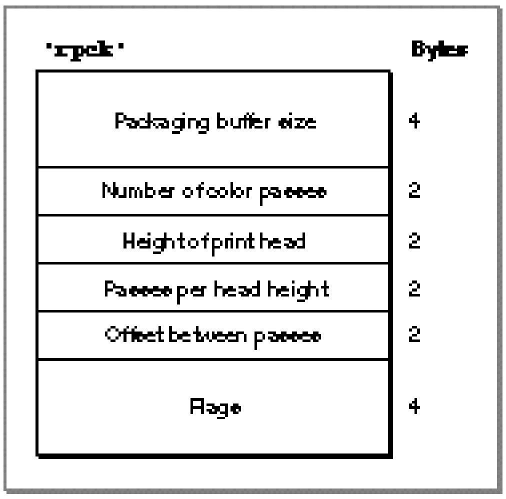
```
resource gxRasterPrefsType (gxRasterPrefsID, sysHeap, purgeable)
{
   gxDefaultRaster,        /* default options are fine */

   0x00900000,0x00900000,  /* 144X144 dpi device */
   16,                     /* min band size is 2 head heights */
   0,                      /* max band size (0 is full page) */
   0x00004000,             /* RAM percentage (25%) */
   100*1024,               /* RAM slop (100K) */
   4,                      /* 4 bit per plane device */
      {
      /* dithering offscreen */
      3,          /* gxDontSetHalftone+gxDotTypeIsDitherLevel */

      0x002D0000,          /* angle unused for dithering */
      0x003C0000,          /* freq unused for dithering */
      4,                   /* dithering with a level of 4 */
      gxLuminanceTint,     /* tint space unused for dithering */

                        /* dot color is unused for dithering */
      gxRGBSpace, gxNoProfile, 0, 0, 0, 0,
                        /* background is unused for dithering */
      gxRGBSpace, gxNoProfile, 0xFFFF, 0xFFFF, 0xFFFF, 0,

      gxRGBSpace,       /* halftone space unused for dithering */

      gxIndexedSpace,         /* indexed color space */
      gxPrintingDriverBaseID, /* ID of the color set to use */
      gxPrintingDriverBaseID, /* ID of the color profile to use */
      };
};
```

The Raster Package ('rpck') Resource

The raster package (`'rpck'`) resource, of type `gxRasterPackType`, controls how bitmap data is packed into rasters for raster drivers. This resource is optional.

Figure 6-24 shows the structure of a raster package resource.

Figure 6-24 The raster package resource



The raster package resource contains the following elements:

- Packaging buffer size. The number of bytes in the packing buffer. This value must be greater than or equal to the maximum number of bytes needed for one head pass on the printer.
- Number of color passes. The number of color passes required by the device to print an image. This value is typically 1 (for monochrome printers) or 4 (for CMYK printers).
- Height of print head. The height of the print head, in pixels.
- Passes per head height. The number of head passes required to achieve the height of the print head.
- Offset between passes. The offset, in pixels, between head passes.

- Flags. The raster package options, which can be combined together into a single value. The constants for the raster packing options are shown in Table 6-32.

Table 6-32 Raster package options

Constant	Value	Explanation
`gxSendAllColors`	0x00000001	Send raster bands even if all clear
`gxInterlaceColor`	0x00000002	Interlace colors because ribbon contamination is of concern
`gxOverlayColor`	0x00000004	This is a color printer that does not have a ribbon contamination problem

The ID of the raster package resource must be the constant `gxRasterPackID`. Listing 6-22 shows an example of a raster package resource for an ImageWriter II printer.

Listing 6-22 An example of a raster package resource

```
resource gxRasterPackType (gxRasterPackID, sysHeap, purgeable)
{
   2500,             /* ImageWriter packing buffer size */
   4,                /* 4 color passes */
   16,               /* print head is 16 pixels high */
   2,                /* requires 2 passes to achieve 16 pixels */
   1,                /* 1 pixel difference between passes */
   gxInterlaceColor  /* use interlace to avoid contamination */
};
```

This resource specifies that the ImageWriter II printer uses a packing buffer that is 2500 bytes long, which is long enough for the largest line packaged by the driver. Since the driver uses CMYK color, 4 passes are required to print an image. The print head is 16 pixels high, and it requires 2 printing passes to print 16 pixels of data.

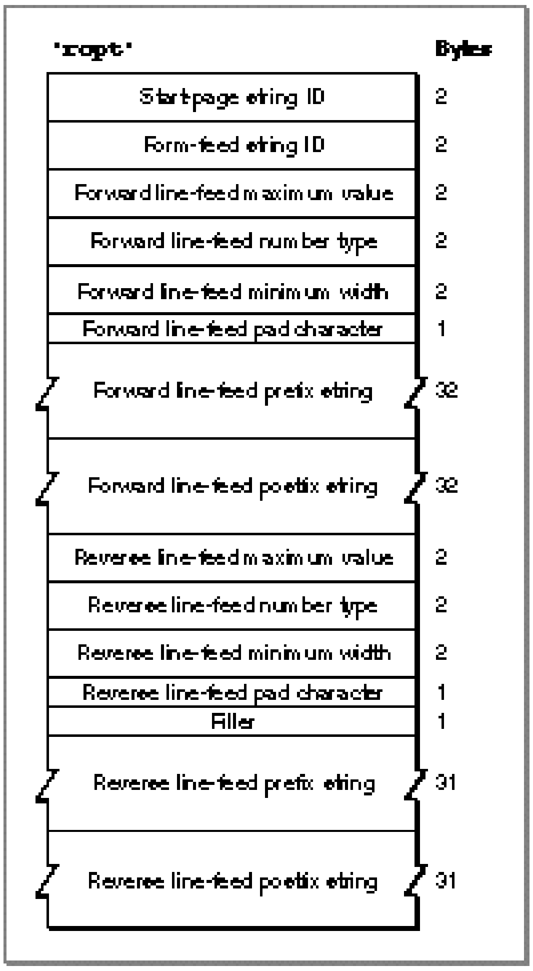
If you use the default implementation of the `GXRasterDataIn` message, you must define a raster package (`'rpck'`) resource. The `GXRasterDataIn` message is described on page 4-97 in the chapter "Printing Messages" in this book.

The Raster Package Controls ('ropt') Resource

The raster package controls (`'ropt'`) resource, of type `gxRasterPackOptionsType`, is used to define how some forms of line feeding are performed on a raster device. This resource is optional.

Figure 6-25 shows the structure of a raster package controls resource.

Figure 6-25 The raster package controls resource


'rcpt'
Bytes
Start-page string ID
2
Form-feed string ID
2
Forward line-feed maximum value
2
Forward line-feed number type
2
Forward line-feed minimum width
2
Forward line-feed pad character
1
Forward line-feed prefix string
32
Forward line-feed postfix string
32
Reverse line-feed maximum value
2
Reverse line-feed number type
2
Reverse line-feed minimum width
2
Reverse line-feed pad character
1
Filler
1
Reverse line-feed prefix string
31
Reverse line-feed postfix string
31

The raster package controls resource contains the following elements:

- Start-page string ID. The ID of the wide string (`'wstr'`) resource that is used as the start-page sequence. Wide strings are strings that require a 16-bit, short integer value to define their length. The length value is stored in the first 2 bytes of the string.
- Form-feed string ID. The ID of the wide string (`'wstr'`) resource that is used as the form-feed sequence.
- Forward line-feed maximum value. The maximum amount of a forward line feed.
- Forward line-feed number type. The type of the number used to specify the forward line feed. Constants for the number types are shown in Table 6-33.
- Forward line-feed minimum width. The minimum width of the forward line feed.
- Forward line-feed pad character. The pad character used for the forward line feed.
- Forward line-feed prefix string. The prefix string used for the forward line feed.

- Forward line-feed postfix string. The postfix string used for the forward line feed.
- Reverse line-feed maximum value. The maximum amount of a reverse line feed.
- Reverse line-feed number type. The type of the number used to specify the reverse line feed. The possible values are shown in Table 6-33.
- Reverse line-feed minimum width. The minimum width of the reverse line feed.
- Reverse line-feed pad character. The pad character used for the reverse line feed.
- Reverse line-feed prefix string. The prefix string used for the reverse line feed.
- Reverse line-feed postfix string. The postfix string used for the reverse line feed.

Table 6-33 shows the constants that you can use to specify the number types used in defining line feeds in raster package controls resources.

Table 6-33 Number types for specifying line feeds

Constant	Value	Explanation
`gxRasterNumNone`	0	The number is not sent at all. Only the prefix and postfix strings are sent.
`gxRasterNumDirect`	1	The number is first padded to minimum width bytes by prepending 0’s to it and is then sent as a sequence of hex bytes.
`gxRasterNumToASCII`	2	The number is converted to ASCII and then front-padded with the specified pad character to make it minimum width bytes long. The ASCII characters are then sent to the printer.

The ID of a raster package controls resource must be the constant `gxRasterPackOptionsID`. Listing 6-23 shows an example of a raster package controls resource for the ImageWriter II printer driver.

Listing 6-23 An example of a raster package controls resource

```
resource gxRasterPackOptionsType (gxRasterPackOptionsID, sysHeap,
                                                       purgeable)
{
   gxPrintingDriverBaseID,       /* ID of start page wstr res */
   gxPrintingDriverBaseID+10,    /* ID of form-feed wstr res */
         /* forward line-feed characteristics */
   98,                     /* max line-feed amount is 98 */
   gxRasterNumToASCII,     /* express line-feed as ASCII */
   2,                      /* minimum width is 2 */
   "0",                    /* pad with zeros */
   "\0X1BT",               /* <esc>T is set line-feed size */
   "\0X1Br\0X0A"  /* <esc>r<lf> is direction reverse, line feed */

                     /* reverse line-feed characteristics */
   98,               /* max line-feed amount is 98 */
   gxRasterNumToASCII,/* express line-feed as ASCII */
   2,                /* minimum width is 2 */
   "0",              /* pad with zeros */
   "\0X1BT",         /* <esc>T is set line-feed size */
   "\0X1Br\0X0A"  /* <esc>r<lf> is direction reverse, line feed */
};

resource 'wstr' (gxPrintingDriverBaseID, sysHeap, purgeable)
{
/*
   Start page string: unidirectional, 144 dpi ==
      ESC 5A 0700 ESC>ESCp
*/
   "\0X1BO" "\0X1B>" "\0X1Bp"
};

resource 'wstr' (gxPrintingDriverBaseID+10, sysHeap,purgeable)
{
      /* End-page string: a control-L for IW's form feed */
   "\0X0C",
};
```

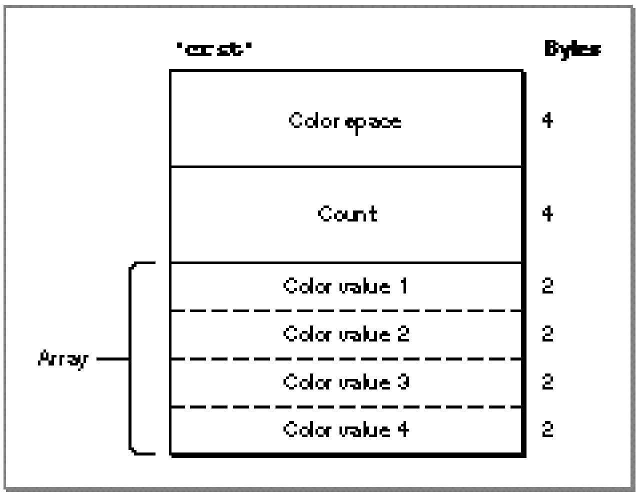
If you use the default implementation of the `GXRasterLineFeed` or `GXRasterPackageBitmap` messages, you must define a raster package controls (`'ropt'`) resource. The `GXRasterLineFeed` and `GXRasterPackageBitmap` messages are described in the chapter "Printing Messages" in this book.

The Color Set ('crst') Resource

The color set (`'crst'`) resource is used with raster imaging drivers to specify a set of replacement strings that are used when interpreting the information returned by a printer. This resource is optional.

Figure 6-26 shows the structure of a color set resource.

Figure 6-26 The color set resource



The color set resource consists of a number of color values that are defined for a certain color space.

- Color space. The color space that the colors are defined for. Use one of the values described in the chapter "Colors and Color-Related Objects" in *Inside Macintosh: QuickDraw GX Objects*.
- Count. The number of color values defined in the array that follows.

Each color entry in the array consists of up to four integer values. What each value specifies depends on the type of color space, as described in the chapter "Colors and Color-Related Objects" in *Inside Macintosh: QuickDraw GX Objects*.

- Color value 1. The first color value.
- Color value 2. The second color value.
- Color value 3. The third color value.
- Color value 4. The fourth color value.

Listing 6-24 shows an example of a color set resource for the ImageWriter II printer driver. This printer uses the RGB color space and defines only 8 colors.

Listing 6-24 An example of a color set resource

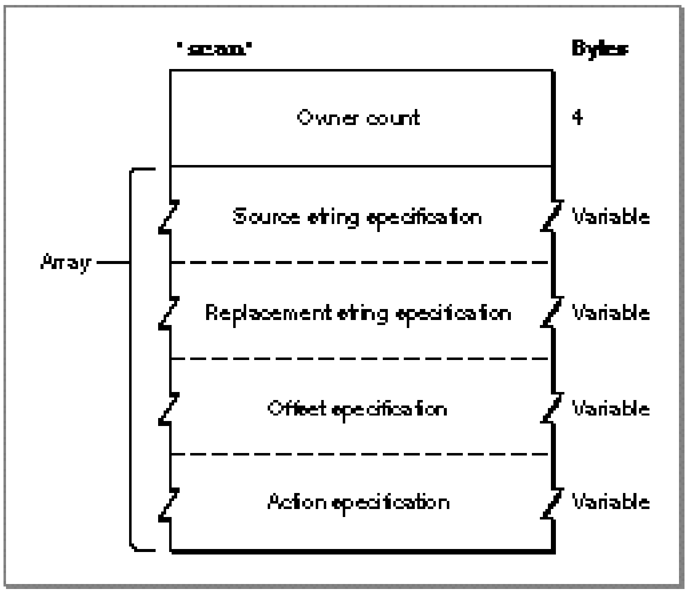
```
resource gxColorSetResType (gxPrintingDriverBaseID, sysHeap,
                                                     purgeable)
{
   gxRGBSpace,
   {
      /* R     G     B     unused */
      0xFFFF,0xFFFF,0xFFFF,0x0000;  /* white */
      0xFFFF,0xFFFF,0x0000,0x0000;  /* yellow */
      0xFFFF,0x0000,0xFFFF,0x0000;  /* magenta */
      0xFFFF,0x0000,0x0000,0x0000;  /* red */
      0x0000,0xFFFF,0xFFFF,0x0000;  /* cyan */
      0x0000,0xFFFF,0x0000,0x0000;  /* green */
      0x0000,0x0000,0xFFFF,0x0000;  /* blue */
      0xFFFF,0xFFFF,0xFFFF,0x0000;  /* white */
      0x0000,0x0000,0x0000,0x0000;  /* black */
   };
};.
```

The PostScript Scanning ('scan') Resource

The PostScript scanning (`'scan'`) resource, of type `gxPostscriptScanningType`, is used with PostScript drivers to specify a set of replacement strings that are used when interpreting the status information returned by a PostScript printer. This resource is optional.

Figure 6-27 shows the structure of a PostScript scanning resource.

Figure 6-27 The PostScript scanning resource



The PostScript scanning resource consists of an owner count field and an array of string specifications.

n Owner count.

Each string specification contains four strings. These strings together form the specification of how to replace strings in the status information that is returned by the printer.

- Source string specification. Specifies the string to be searched for, as described in Table 6-34.
- Replacement string specification. Specifies the string with which to replace the source string.
- Offset specification. Specifies the offset at which to make the replacement, as described in Table 6-35.
- Action specification. Specifies the action to take after making the replacement, as described in Table 6-36.

You define the source string and replacement strings as `StringScan` specifications, which have five variations, as shown in Table 6-34.

Table 6-34 Scan string specifications for the PostScript scanning resource

String type	Value	Explanation	Parameters
`SimpleScan`	0	The parameter defines the string to uses	A `wide` string
`UserNameScan`	1	Uses the name of the user as the string	None
`DocumentNameScan`	2	Uses the name of the document as the string	None
`PrinterNameScan`	3	Uses the name of the printer as the string	None
`NilPtrScan`	4	Uses the empty string	None

You define the offset with an `OffsetScan` specification, which has six variations, as shown in Table 6-35. None of these requires the use of parameters.

Table 6-35 Scan offset specifications for the PostScript scanning resource

Offset type	Value	Explanation
`SimpleOffset`	0	No offset used–start at the beginning of the buffer
`SameAsPreviousOffset`	1	Used to insert text immediately before a source string
`ReturnedOffset`	2	Used to insert text immediately after a source string

Table 6-35 Scan offset specifications for the PostScript scanning resource (continued)

Offset type	Value	Explanation
`SimpleRepeat`	16	Same as `SimpleOffset` and the replacement repeats for all occurences of the source string
`SampleAsPreviousRepeat`	17	Same as `SameAsPreviousOffset` and repeats for all occurrences of the source string
`ReturnedRepeat`	18	Same as `ReturnedOffset` and repeats for all occurrences of the source string

You define the action with an `ActionScan` specification, which has two variations, as shown in Table 6-36.

Table 6-36 Scan-action specifications for the PostScript scanning resource

Action type	Value	Explanation	Parameters
`NoAction`	0	No action taken	None
`SimpleAction`	1	If an error occurs, a printing alert box is displayed	Two integers The first defines the type of error to generate (`normal`, `nonFatalError`, or `fatalError`) The other specifies the resource ID of the printing alert to use

Listing 6-25 shows the PostScript scanning resources from the Apple LaserWriter IIg printer driver.

Listing 6-25 An example of PostScript scanning resources for a printer driver

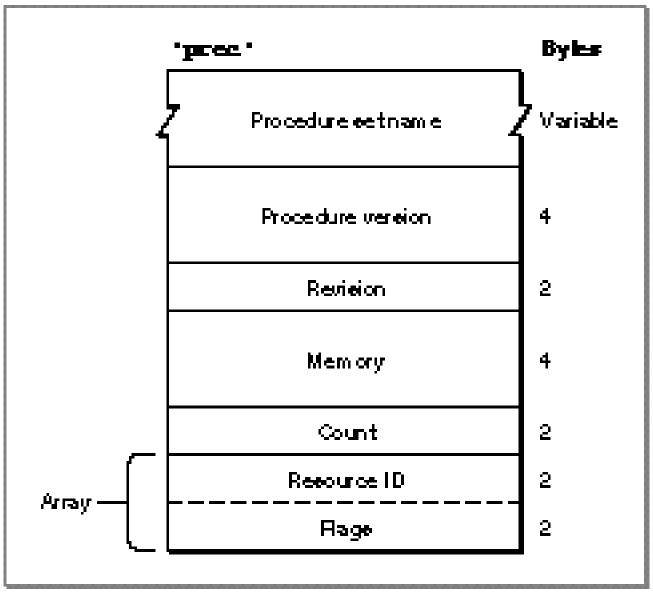
```
resource gxPostscriptScanningType (gxPostscriptScanningID,
                                              sysHeap, purgeable)
{
   0,
   {
   SimpleScan {"busy"}, SimpleScan { "printer busy" },
      SimpleOffset {}, NoAction {};
   SimpleScan {"waiting"}, SimpleScan {"preparing data"},
      SimpleOffset {}, NoAction {};
   SimpleScan {"job:"}, SimpleScan {"User"},
      SimpleOffset {}, NoAction {};
   }
};

resource gxPostscriptScanningType (gxPostscriptScanningID + 1,
                                          sysHeap, purgeable)
{
   0,
   {
   SimpleScan {"%%["}, SimpleScan {""},
      SimpleOffset {}, NoAction {};
   SimpleScan {"]%%"}, SimpleScan {""},
      SimpleOffset {}, NoAction {};
   SimpleScan {"PrinterError"}, SimpleScan {"Printer"},
      SimpleOffset {}, NoAction {};
   };
};
```

The PostScript Procedure Set Control ('prec') Resource

The PostScript procedure set control (`'prec'`) resource, of type `gxPostscriptProcSetControlType`, is used with PostScript drivers to define a procedure set that can be downloaded to a PostScript printer. This resource is optional.

Figure 6-28 shows the structure of a procedure set control resource.

Figure 6-28 The PostScript procedure set control resource



The procedure set control resource contains naming, version, and memory information about the procedure set, followed by the IDs of the resources that comprise the procedure set.

- Procedure set name. The Pascal string name of the procedure set.
- Procedure set version. The procedure set type. This is a fixed version number.
- Revision. The revision number (an integer) of this procedure set.
- Memory usage. The amount of printer memory required for the procedure set.
- Count. The number of resources that comprise the procedure set.

Each procedure in the set is defined in a resource. Each entry in the array consists of the ID of the procedure resource and a flag that specifies what to do with the procedure data.

- Resource ID. The ID of the resource containing a PostScript procedure.
- Flags. A value that describes what to do with the data in the resource. The constants that you can use in this field are shown in Table 6-37.

Table 6-37 PostScript procedure data flags

Constant	Value	Explanation
`donothing`	0	The imaging system should send the entire contents of this resource to the output device without modification.
`dumpwidestring`	1	The imaging system needs to treat the first two bytes (the first short) of the resource as the number of bytes to send to the printer. That many bytes, starting at the third byte, are then sent to the output device without modification.
`dumpstringlist`	2	The imaging system needs to treat the resource like a string list (`'STR#'`) resource. The imaging system dumps each string after appending a line feed and carriage return to the end of each.
`converttohex`	0x0100	This is a modifier that specifies that the imaging system needs to convert the data to hexadecimal format before sending it to the output device. You can specify this modifier with the `donothing` or `dumpwidestring` flags; however, you cannot use it in conjunction with the `dumpstringlist` flag.

The string list resource is described in *Inside Macintosh: Macintosh Toolbox Essentials*.

The PostScript Printer Font Type ('pfnt') Resource

Each PostScript printer font type (`'pfnt'`) resource defines parameters for a specific kind of font that you can use with a PostScript device:

- Adobe™ character set
- Apple character set
- Equivalent character set
- Encoded font

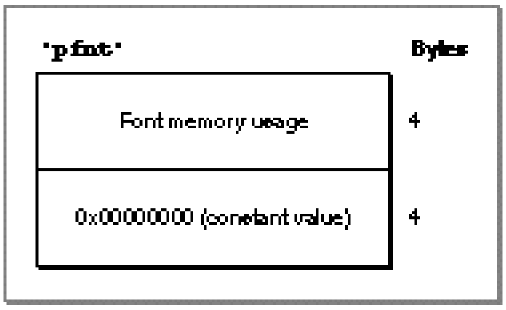
The resources for these kinds of fonts are described in the following sections.

The PostScript Printer Font Type ('pfnt') Resource for the Adobe Character Set

The PostScript printer font type (`'pfnt'`) resource for the Adobe character set is used with PostScript drivers to specify information about fonts that use the Adobe character set. This resource is optional.

Figure 6-29 shows the structure of a PostScript printer font type resource for the Adobe character set.

Figure 6-29 The PostScript printer font type resource for the Adobe character set



The PostScript printer font type resource for the Adobe character set specifies the memory used by the font and contains a constant value.

- Font memory usage. The amount of memory used for the font.
- The constant value for this resource is 0x00000000.

Listing 6-26 shows several of the PostScript printer font type resources for the Adobe character set from the Apple LaserWriter printer driver.

Listing 6-26 Examples of PostScript printer font type resources for the Adobe character set

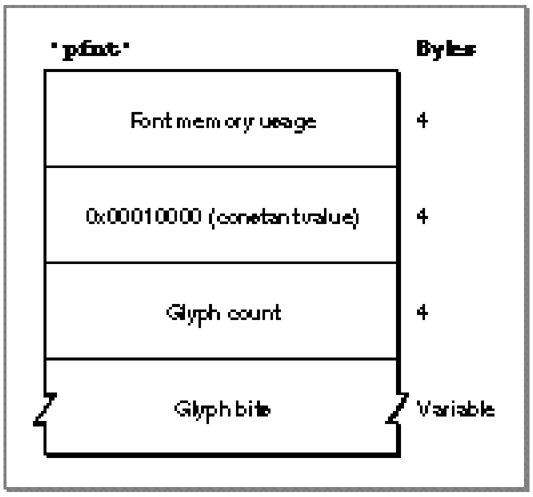
```
resource gxPostscriptPrinterFontType (gxPrintingDriverBaseID + 4,
                                    "Helvetica", sysHeap, purgeable)
{
   ROMFont,
   AdobeCharacterSet {};
};

resource gxPostscriptPrinterFontType (gxPrintingDriverBaseID + 5,
                           "Helvetica-Bold", sysHeap, purgeable)
{
   ROMFont,
   AdobeCharacterSet {};
};

resource gxPostscriptPrinterFontType (gxPrintingDriverBaseID + 6,
                     "Helvetica-BoldOblique", sysHeap, purgeable)
{
   ROMFont,
   AdobeCharacterSet {};
};
```

The PostScript Printer Font Type ('pfnt') Resource for the Apple Character Set

The PostScript printer font type (`'pfnt'`) resource for the Apple character set is used with PostScript drivers to specify information about fonts that use the Apple character set. This resource is optional.

Figure 6-30 shows the structure of a PostScript printer font type resource for the Apple character set.

Figure 6-30 The PostScript printer font type resource for the Apple character set



The PostScript printer font type resource for the Apple character set specifies the memory used by the font, a constant value, and glyph data.

- Font memory usage. The amount of memory used for the font.
- The constant value for this resource is 0x00010000.
- Glyph count. The number of glyphs defined in this resource.
- Glyph bits. The data for the glyphs of this font.

Listing 6-27 shows a PostScript printer font type resource for the Apple character set from the Apple LaserWriter printer driver.

Listing 6-27 An example of a PostScript printer font type resource for the Apple character set

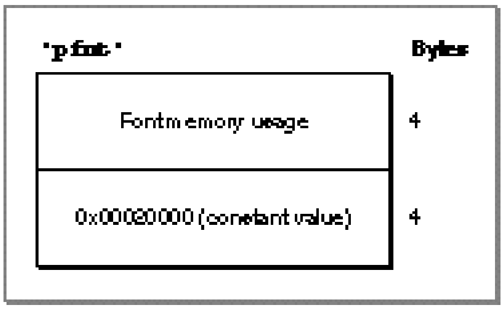
```
resource gxPostscriptPrinterFontType (gxPrintingDriverBaseID + 7,
                                     "Symbol", sysHeap, purgeable)

{
   ROMFont,
   AppleCharacterSet {
      191,
      $"9FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
   };
};
```

The PostScript Printer Font Type ('pfnt') Resource for the Equivalent Character Set

The PostScript printer font type (`'pfnt'`) resource for the equivalent character set is used with PostScript drivers to specify information about a font that uses the equivalent character set. This resource is optional.

Figure 6-31 shows the structure of a PostScript printer front type resource for an equivalent character set.

Figure 6-31 The PostScript printer font type resource for an equivalent character set



The PostScript printer font type resource for an equivalent character set specifies the memory used by the font and contains a constant value.

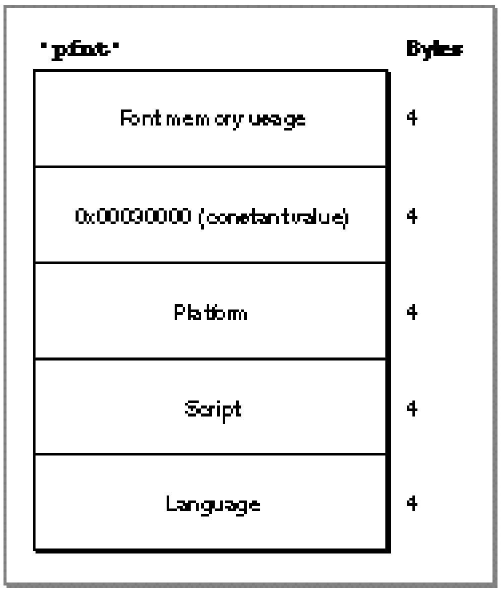
- Font memory usage. The amount of memory used for the font.
- The constant value for this resource is 0x00020000.

The PostScript Printer Font Type ('pfnt') Resource for an Encoded Font

The PostScript printer font type (`'pfnt'`) resource for an encoded font is used with PostScript drivers to specify information about encoded fonts. This resource is optional.

Figure 6-32 shows the structure of a PostScript printer font type resource for an encoded font.

Figure 6-32 The PostScript printer font type resource for an encoded font



The PostScript printer font type resource for an encoded font specifies the memory used by the font, a constant value, and information about the platform, script, and language of the font. The platform, script, and language values used in this resource are described in the chapter “Fonts” in *Inside Macintosh: QuickDraw GX Typography.*

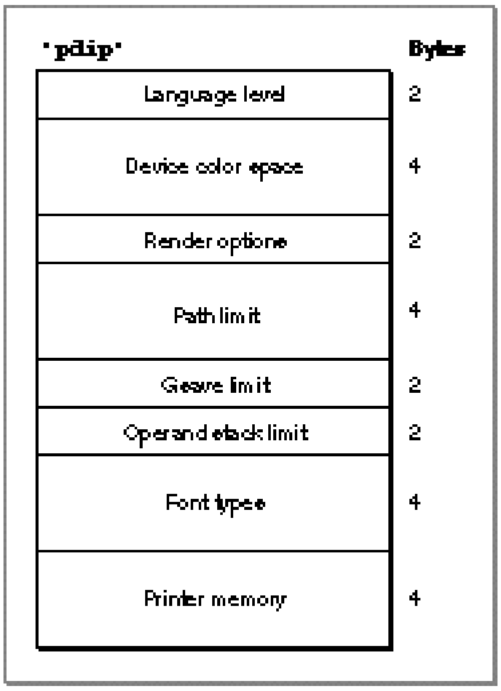
- Font memory usage. The amount of memory used for the font.
- The constant value for this resource is 0x00030000.
- Platform. The platform ID for this font.
- Script. The ID of the script system for this font.
- Language. The language ID for this font.

The PostScript Preferences ('pdip') Resource

The PostScript preferences (`'pdip'`) resource, of type `gxPostscriptPrefsType`, is used with PostScript drivers to specify information about a specific PostScript device. This resource is optional.

Figure 6-33 shows the structure of a PostScript preferences resource.

Figure 6-33 PostScript preferences resource



The PostScript preferences resource describes a number of attributes of the printer:

- Language level. The PostScript language level supported by the printer.
- Device color space. The color space supported by the printer. The color-space value constants are shown in the chapter “Colors and Color-Related Objects” in *Inside Macintosh: QuickDraw GX Objects.* The values that are valid for this field are `gxGraySpace`, `gxCMYKSpace`, and `gxRGBSpace`.
- Render options. Rendering options for the PostScript printer. This value is the combined value of the constants you that you include from the choices shown in Table 6-38.
- Path limit. The largest number of points that can be active in the device during the imaging process.
- Gsave limit. The greatest number of gsaves that can be performed on the device.

- Operand stack limit. The maximum number of objects that can be placed in the stack at any one time.
- Font types. The stream types that the imaging system needs to use when downloading a font. The possible values are shown in Table 6-39.
- Printer memory. The amount of memory available in the printer.

Table 6-38 shows the constants that you can use to define the render options field of the PostScript preferences resource.

Table 6-38 PostScript render options

Constant	Value	Explanation
`gxNeedsAsciiOption`	1	The driver needs to convert all binary data to 7-bit ASCII values.
`gxNeedsCommentsOption`	2	The driver needs to issue PostScript comments.
`gxBoundingBoxesOption`	4	The driver needs to calculate values for the `%%BoundingBox:` and `%%PageBoundingBox:` variables. This option implies the `gxNeedsCommentsOption` constant.
`gxPortablePostScriptOption`	8	The driver needs to generate PostScript that is not device specific.
`gxUseLevel2ColorOption`	128	The driver is to use Level 2 device-independent color when printing to a Level 2 output device.

Table 6-39 shows the values that you can specify in the font types field of the PostScript preferences resource.

Table 6-39 Font stream types

Constant	Value	Explanation
`trueTypeStreamType`	0x0001	TrueType font format
`type1StreamType`	0x0002	PostScript Type 1 font format
`type3StreamType`	0x0004	PostScript Type 3 font format
`type42StreamType`	0x0008	PostScript Type 4.2 font format
`type42GXStreamType`	0x0010	GX PostScript Type 4.2 font format
`portableStreamType`	0x0020	Portable font format
`flattenedStreamType`	0x0040	Flattened font format

Summary of Printing Resources

C Summary

Constants and Data Types

```
   /* basic client types */
#define gxPrintingManagerType 'pmgr'
#define gxPrinterDriverType   'pdvr'
#define gxPrintingExtensionType'pext'
#define gxAnyPrinterType      'univ'
#define gxPortableDocPrinterType 'gxpd'


#define gxRasterPrinterType   'rast'
#define gxPostscriptPrinterType'post'
#define gxVectorPrinterType   'vect'

#define gxPrintingTagID (-28672) /* gxTag ID for printing collections */



/*
   The following constants are for resources used by both extensions and
   drivers.
*/

      /* base IDs for extension & driver resources */
#define gxPrintingDriverBaseID (-28672)
#define gxPrintingExtensionBaseID (-27136)

#define gxOverrideType  'over'      /* override resource type */

#define gxDriverUniversalOverrideID       (gxPrintingDriverBaseID)
#define gxDriverImagingOverrideID         (gxPrintingDriverBaseID + 1)
#define gxDriverCompatibilityOverrideID   (gxPrintingDriverBaseID + 2)

#define gxExtensionUniversalOverrideID          gxPrintingExtensionBaseID
#define gxExtensionImagingOverrideSelectorID    gxPrintingExtensionBaseID
```

```
/*
   The following constants are used to identify printing messages for use
   in both extensions and drivers.
*/

         /* identifiers for universal message overrides... */

#define gxInitialize             0
#define gxShutDown               1

#define gxJobIdle                2
#define gxJobStatus              3
#define gxPrintingEvent          4

#define gxJobFormatDialog        5
#define gxFormatDialog           6
#define gxJobPrintDialog         7
#define gxFilterPanelEvent       8
#define gxHandlePanelEvent       9
#define gxParsePageRange         10

#define gxDefaultJob             11
#define gxDefaultFormat          12
#define gxDefaultPaperType       13
#define gxDefaultPrinter         14

#define gxCreateSpoolFile        15
#define gxSpoolPage              16
#define gxSpoolData              17
#define gxSpoolResource          18
#define gxCompleteSpoolFile      19

#define gxCountPages             20
#define gxDespoolPage            21
#define gxDespoolData            22
#define gxDespoolResource        23
#define gxCloseSpoolFile         24

#define gxStartJob               25
#define gxFinishJob              26
#define gxStartPage              27
#define gxFinishPage             28
```

```
#define gxPrintPage              29

#define gxSetupImageData         30
#define gxImageJob               31
#define gxImageDocument          32
#define gxImagePage              33
#define gxRenderPage             34
#define gxCreateImageFile        35

#define gxOpenConnection         36
#define gxCloseConnection        37
#define gxStartSendPage          38
#define gxFinishSendPage         39

#define gxWriteData              40
#define gxBufferData             41
#define gxDumpBuffer             42
#define gxFreeBuffer             43

#define gxCheckStatus            44
#define getDeviceStatus          45

#define gxFetchTaggedData        46

#define gxGetDTPMenuList         47
#define gxDTPMenuSelect          48
#define gxDTPHandleAlertFilter   49

#define gxJobFormatModeQuery     50

#define gxWriteStatusToDTPWindow 51
#define gxInitializeStatusAlert  52
#define gxHandleAlertStatus      53
#define gxHandleAlertEvent       54

#define gxCleanupStartJob        55
#define gxCleanupStartPage       56
#define gxCleanupOpenConnection  57
#define gxCleanupStartSendPage   58

#define gxDefaultDesktopPrinter  59
#define gxCaptureOutputDevice    60
```

```
#define gxOpenConnectionRetry    61
#define gxExamineSpoolFile       62

#define gxFinishSendPlane        63
#define gxDoesPaperFit           64
#define gxChooserMessage         65

#define gxFindPrinterProfile     66
#define gxFindFormatProfile      67
#define gxSetPrinterProfile      68
#define gxSetFormatProfile       69

      /* identifiers for Macintosh Printing Manager message overrides */
#define gxPrOpenDoc              0
#define gxPrCloseDoc             1
#define gxPrOpenPage             2
#define gxPrClosePage            3
#define gxPrintDefault           4
#define gxPrStlDialog            5
#define gxPrJobDialog            6
#define gxPrStlInit              7
#define gxPrJobInit              8
#define gxPrDlgMain              9
#define gxPrValidate             10
#define gxPrJobMerge             11
#define gxPrGeneral              12
#define gxConvertPrintRecordTo   13
#define gxConvertPrintRecordFrom 14
#define gxPrintRecordToJob       15

   /* identifiers for raster imaging message overrides */
#define gxRasterDataIn           0
#define gxRasterLineFeed         1
#define gxRasterPackageBitmap    2

   /* identifiers for PostScript imaging message overrides */
#define gxPostscriptQueryPrinter              0
#define gxPostscriptInitializePrinter         1
#define gxPostscriptResetPrinter              2
#define gxPostscriptExitServer                3
#define gxPostscriptGetStatusText             4
#define gxPostscriptGetPrinterText            5
#define gxPostscriptScanStatusText            6
#define gxPostscriptScanPrinterText           7
```

```
#define gxPostscriptGetDocumentProcSetList      8
#define gxPostscriptDownloadProcSetList         9
#define gxPostscriptGetPrinterGlyphsInformation 10
#define gxPostscriptStreamFont                  11
#define gxPostscriptDoDocumentHeader            12
#define gxPostscriptDoDocumentSetUp             13
#define gxPostscriptDoDocumentTrailer           14
#define gxPostscriptDoPageSetUp                 15
#define gxPostscriptSelectPaperType             16
#define gxPostscriptDoPageTrailer               17
#define gxPostscriptEjectPage                   18
#define gxPostscriptProcessShape                19

   /* identifiers for vector imaging message overrides */
#define gxVectorPackageData         0
#define gxVectorLoadPens            1
#define gxVectorVectorizeShape      2

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

/*
   The following resource types and IDs are used by extensions.
*/
```

```
#define gxExtensionScopeType  'scop'
#define gxDriverScopeID             gxPrintingExtensionBaseID
#define gxPrinterScopeID            gxPrintingExtensionBaseID+1
#define gxPrinterExceptionScopeID   gxPrintingExtensionBaseID+2

#define gxExtensionLoadType'load'
#define gxExtensionLoadID        gxPrintingExtensionBaseID

#define gxExtensionLoadFirst  0x00000100
#define gxExtensionLoadAnywhere0x7FFFFFFF
#define gxExtensionLoadLast   0xFFFFFF00

#define gxExtensionOptimizationType'eopt'
#define gxExtensionOptimizationID   gxPrintingExtensionBaseID

      /* extension optimization values*/

#define gxExecuteDuringImaging         TRUE
#define gxNeedDeviceStatus             TRUE
#define gxChangePageAtGXDespoolPage    TRUE
#define gxChangePageAtGXImagePage      TRUE
#define gxChangePageAtGXRenderPage     TRUE
#define serverPresenceRequired         FALSE
#define clientPresenceRequired         FALSE
#define dontexecuteDuringImaging       FALSE
#define dontneedDeviceStatus           FALSE
#define dontchangePageAtDespoolPage    FALSE
#define dontchangePageAtImagePage      FALSE
#define dontchangePageAtRenderPage     FALSE
#define notServerPresenceRequired      FALSE
#define notClientPresenceRequired      FALSE

/*
   The following resource types and IDs are used by writers of printer
   drivers.
*/

   /* imaging Resources */
#define gxImagingSystemSelectorType'isys'
#define gxImagingSystemSelectorID(gxPrintingDriverBaseID)

   /* raster rendering preferences resources */
#define gxRasterPrefsType     'rdip'
#define gxRasterPrefsID       gxPrintingDriverBaseID
```

```
   /* resource type for specifiying a color set*/
#define gxColorSetResType     'crst'

   /* resource type and ID for raster generic driver packaging preferences */
#define gxRasterPackType      'rpck'
#define gxRasterPackID        gxPrintingDriverBaseID

   /* resource type and ID for raster generic driver packaging options */
#define gxRasterNumNone    0  /* number isn't output at all */
#define gxRasterNumDirect  1  /* lowest minWidth bytes as data */
#define gxRasterNumToASCII 2  /* minWidth ASCII characters */

#define gxRasterPackOptionsType'ropt'
#define gxRasterPackOptionsID    gxPrintingDriverBaseID

   /* resource type for the PostScript procedure set control resource */

#define gxPostscriptProcSetControlType 'prec'

   /* resource type for the PostScript printer gxFont resource */
#define gxPostscriptPrinterFontType 'pfnt'

   /* resource type and id for the PostScript imaging preferences */
#define gxPostscriptPrefsType    'pdip'
#define gxPostscriptPrefsID      gxPrintingDriverBaseID

   /* resource type and id for the PostScript default scanning code */
#define gxPostscriptScanningType 'scan'
#define gxPostscriptScanningID0

   /* resource for type for color matching */
#define gxColorMatchingDataType  'prof'
#define gxColorMatchingDataID    gxPrintingDriverBaseID

   /* resource type and id for the default bin and paper specifications */
#define gxTrayCountDataType   'tray'
#define gxTrayCountDataID     gxPrintingDriverBaseID

/*
   The following resource types and IDs are used to define input and output
   parameters for printer drivers.
*/

   /* resource type and ID for default IO and buffering resources */
#define gxUniveralIOPrefsType 'iobm'
#define gxUniversalIOPrefsID  gxPrintingDriverBaseID
```

```
   /* resource defines for default implementation of */
   /*  GXCaptureOutputDevice, which only handles PAP devices */
#define gxCaptureType            'cpts'
#define gxCaptureStringID        (gxPrintingDriverBaseID)
#define gxReleaseStringID        (gxPrintingDriverBaseID + 1)
#define gxUncapturedAppleTalkType(gxPrintingDriverBaseID + 2)
#define gxCapturedAppleTalkType  (gxPrintingDriverBaseID + 3)

   /* resource type and ID for driver papertypes in individual files */
#define gxSignatureType       'sig '
#define gxPapertypeSignatureID   0

   /* file type for driver papertypes placed in individual files */
#define gxDrvrPaperType    'drpt'

/*
   The following resource types and IDs are used to support Macintosh
   Printing Manager compatibility.
*/

#define gxCustType   'cust'
#define gxCustID     -8192

#define gxReslType 'resl'
#define gxReslID  -8192
#define gxDiscreteResolution 0


#define gxStlDialogResID -8192
#define gxJobDialogResID -8191
```

The Buffering and I/O Preferences Structure

```
struct gxIoPrefsRec {
   unsigned long  communicationsOptions;  /* options for input & output */
   unsigned long  numBuffers;             /* number of buffers to allocate */
   unsigned long  bufferSize;             /* number of bytes per buffer */
   unsigned long  numReqBlocks;           /* number of I/O request blocks */
   unsigned long  openCloseTimeout;       /* timeout for open and close */
   unsigned long  readWriteTimeout        /* timeout for read and write */
};

typedef struct gxIOPrefsRec gxIOPrefsRec, *gxIOPrefsPtr, **gxIOPrefsHdl;
```

The Customization Structure

```
struct gxCustomizationRec {
   short horizontalResolution;   /* horizontal resolution */
   short verticalResolution;     /* vertical resolution */
   short upDriverType;           /* type of Macintosh Printing Manager
                                     interface to use */
   Point patternStretch;         /* pattern stretching factor */
   short translatorSettings      /* settings for translator */
};

typedef struct gxCustomizationRec gxCustomizationRec, *gxCustomizationPtr,
**gxCustomizationHdl;
```

The Resolution Structure

```
struct gxResolutionRec {
   short    rangeType;           /* always 1 */
   short    xMinimumResolution;
   short    xMaximumResolution;
   short    yMinimumResolution;
   short    yMaximumResolution;
   short    resolutionCount;
   Point    resolutions[1];      /* array of points */
};

typedef struct gxResolutionRec gxResolutionRec, *gxResolutionPtr,
**gxResolutionHdl;
```

Raster Preferences Structure

```
struct gxRasterPrefsRec {
   gxRasterRenderOptions
               renderOptions;    /* raster imaging options */
   Fixed       hImageRes;        /* horiz imaging resolution */
   Fixed       vImageRes;        /* vert imaging resolution */
   short       minBandSize;      /* minimum band size to use */
   short       maxBandSize;      /* maximum band size to use */
   Fixed       ramPercentage;    /* maximum percentage of RAM to use */
   long        ramSlop;          /* minimum RAM to leave free */
   short       depth;            /* depth, in pixels, per plane*/
   short       numPlanes;        /* number of planes to render */
   gxPlaneSetupRec
               planeSetup[1];    /* one for each plane */
};

typedef struct gxRasterPrefsRec gxRasterPrefsRec, *gxRasterPrefsPtr,
**gxRasterPrefsHdl;
```

Raster Render Options

```
typedef long gxRasterRenderOptions;

enum {
   gxDefaultRaster   = 0x00000000,  /* default options */
   gxDontResolveTransferModes
                     = 0x00000001,  /* 0 means resolve, 1 means don’t */
   gxRenderInReverse = 0x00000002,  /* traverse in reverse */
   gxOnePlaneAtATime = 0x00000004,  /* render each separately */
   gxSendAllBands    = 0x00000008   /* send all bands, even if empty */
};
```

Raster Package Structure

```
struct gxRasterPackageRec {
   Ptr      bufferSize;    /* buffer size of packaging */
   short    colorPasses;   /* number of color passes */
   short    headHeight;    /* height of print head in pixels */
   short    numberPasses;  /* number of passes per head height */
   short    passOffset;    /* offset between passes, in pixels */
   gxRasterPackgeOptions
            packageOptions;/* packaging options */
};

typedef struct gxRasterPackageRec gxRasterPackageRec, *gxRasterPackagePtr,
**gxRasterPackageHdl;
```

Raster Package Options

```
enum {      /* bit fields in gxRasterPackageOptions */
   gxSendAllColors   = 0x00000001,  /* send all bands, even if empty */
   gxInterlaceColor  = 0x00000002,  /* ribbon contamination */
   gxOverlayColor    = 0x00000004,  /* no ribbon problem */
   gxUseColor        = (gxInterlaceColor|gxOverlayColor);
};

typedef long gxRasterPackageOptions;
```

Raster Package Controls Structure

```
struct gxRasterPackageControlsRec {
   short    startPageStringID;   /* ID of string to send at start of page */
   short    formFeedStringID;    /* ID of string to send for form feed */
   short    forwardMax;          /* maximum amount of forward line feed */
   gxStandardNumberRec
            forwardLineFeed;     /* number struct to express line feed */
   short    reverseMax;          /* maximum amount of reverse line feed */
   gxStandardNumberRec
            reverseLineFeed;     /* number struct to express reverse line
                                     feed */
};

typedef struct gxRasterPackageControlsRec gxRasterPackageControlsRec,
*gxRasterPackageControlsPtr, **gxRasterPackageControlsHdl;
```

Standard Number Structure

```
struct gxStandardNumberRec {
   short    numberType;    /* type of numeric output desired */
   short    minWidth;      /* minimum output width of number */
   char     padChar;       /* pad character */
   char     alignment;
   Str31    startString;   /* the prefix string */
   Str31    endString;     /* the postfix string */
};

typedef struct gxStandardNumberRec gxStandardNumberRec, *gxStandardNumberPtr;
```

Glossary

application phase In QuickDraw GX printing, the phase when the application calls QuickDraw GX and interacts with the user by displaying dialog boxes to establish printing parameters, such as page orientation and paper type.

background task A process that runs concurrently with another process without being the primary focus of the user's attention. The background task is allocated a percentage of the total processor time to accomplish its tasks. Several background tasks can be active at the same time. Compare **foreground task.**

default implementation The implementation of a printing message that is provided by QuickDraw GX. This is the code that executes if your printing extension or printer driver does not totally override the message.

despooling In QuickDraw GX printing, the process during the imaging phase of printing during which each previously spooled page is read from the spool file. See also **imaging phase.**

device communications phase In QuickDraw GX printing, the phase when the data that represents the rendered form of each page is sent to the output device. A printing extension or printer driver can only communicate with the printing device during this phase of printing.

foreground task The process that is currently the main task being executed by the system. This generally corresponds to the application that owns the frontmost window on the user's screen. There is only one foreground task at any given time.

forward To pass a message on to the next message handler in a message chain. See also **message chain, message handler, override.**

imaging phase In QuickDraw GX printing, the phase when each previously spooled page is rendered into a form that can be printed on the output device. The imaging phase is composed of two processes: despooling and rendering. See also **despooling** and **rendering.**

imaging system A part of the QuickDraw GX printing software that manages the conversion of QuickDraw GX shapes into data for a specific type of output device, including raster, vector, and PostScript printing devices. When the output device is a printing device, also referred to as a *print imaging system*. See also **raster imaging system, vector imaging system,** and **PostScript imaging system.**

message A notice sent by one message handler to another that a certain condition has arisen or that a certain task needs to be accomplished. See also **printing message.**

message chain A hierarchy of message handlers eligible to receive and respond to messages.

message handler A recipient of messages. In QuickDraw GX printing, applications, printing extensions, printer drivers, and QuickDraw GX are all message handlers, which are part of a message chain.

message override The response, by a message handler, of intercepting a message and taking some action. The response to a message is performed by an override function. See also **override function.**

message-passing architecture A software system driven by messages that are sent in response to certain conditions or events. The messages activate message handlers, which take action in response to the messages. QuickDraw GX printing uses a message-passing architecture.

override (n.) See **message override** and **override function.** (v.) To intercept a message and take action on it.

override function The code, defined in a message handler, that responds to a message. The formats of the functions that override specific printing messages are given in this book. See also **message override.**

partial override An implementation of a printing message override that forwards the message to other message handlers. You typically forward the message at the beginning or end of your override function.

PostScript imaging system The imaging system provided by QuickDraw GX that converts QuickDraw GX shapes into PostScript instructions and data for PostScript output devices such as the Apple LaserWriter family of printers.

printer driver A program that converts data that is sent by an application program into data and control sequences intended for a specific output device.

print imaging system See **imaging system.**

printing alert box An alert box used by QuickDraw GX printing to display information to the user that must be responded to. The alert box is like a dialog box in that it can contain control items. The user must explicitly dismiss an alert box to remove it from the screen.

printing extension An add-on software module that allows you to extend printing functionality provided by applications and printer drivers.

printing message A notice that QuickDraw GX sends to the message handlers in a message chain that a certain printing-related condition has arisen or that a certain printing-related task needs to be accomplished. See also **message chain** and **message handler.**

printing message override See **message override**.

raster imaging system The imaging system provided by QuickDraw GX that converts QuickDraw GX shapes into data and control sequences for raster output devices such as the Apple ImageWriter family of printers.

rendering In QuickDraw GX printing, the process during the imaging phase of printing during which each despooled page is converted into image data that can be printed by the output device. See also **imaging phase.**

spooling phase In QuickDraw GX printing, the phase when the application sends the document pages to disk, in preparation for printing. The printer driver stores printable output in a file from which it is subsequently despooled, rendered, and sent to the output device. See also **despooling.**

total override An implementation of a printing message override that does not forward the message to other message handlers.

vector imaging system The imaging system provided by QuickDraw GX that converts QuickDraw GX shapes into data and control sequences for vector output devices such as graphic plotters.

Index

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and FrameMaker software. Proof pages were created on an Apple LaserWriter Pro printer. Final page negatives were output directly from text files on an Optrotech SPrint 220 imagesetter. Line art was created using Adobe™ Illustrator and Adobe Photoshop. PostScript™, the page-description language for the LaserWriter, was developed by Adobe Systems Incorporated.

Text type is Palatino® and display type is Helvetica®. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Apple Courier.

WRITER
Gary Hillerson

DEVELOPMENTAL EDITOR
George Truett

ILLUSTRATOR
Lisa Hymel

PRODUCTION EDITOR
Gerri Gray

PROJECT MANAGER
Trish Eastman

LEAD WRITER
David Bice

LEAD EDITOR
Laurel Rezeau

ART DIRECTOR / COVER DESIGNER
Barbara Smyth

Special thanks to Ruth Anderson, Hugo Ayala, Tom Dowdy, Dave Hersey, Ken Hittleman, Dan Lipton, Ron Voss

Acknowledgments to Phil Gold, Marq Laube, Gary McCue, Diane Patterson, Rich Pettijohn, Laine Rapin