

Core Printing Features

This chapter describes how your application can use the core set of QuickDraw GX printing features to print documents created with QuickDraw GX. Read the information in this chapter if you want to print your application's documents to an output device. For example, you might use QuickDraw GX to print to a LaserWriter a document that contains some text and a few illustrations.

Before reading this chapter, you should be familiar with the basic concepts and user interface for printing with QuickDraw GX, as described in the chapter "Introduction to Printing With QuickDraw GX" in this book.

This chapter describes the basic QuickDraw GX print objects: a job, a format, and a paper type. This chapter also shows you how to

- set up the QuickDraw GX printing environment
- create a job object that contains the information needed to print a document
- detect error conditions
- print your application's documents
- save job object information when a user saves a document
- dispose of a job object when a user closes a document
- retrieve job object information when a user opens a document
- obtain information on a format object
- display QuickDraw GX print dialog boxes
- support printing from the Finder
- convert a print record into a job object to print existing documents designed for printing with the Macintosh Printing Manager

About Core Printing Features

Core printing features are features that must be implemented to allow printing documents that contain QuickDraw GX graphics or typographical shapes. These features include the ability to print to desktop printers, format a document for a particular printer (a formatting printer), yet allow printing to another printer (the output printer) without reformatting the document. Core features also include the ability to print from the Finder and to print existing documents designed for printing with the Macintosh Printing Manager.

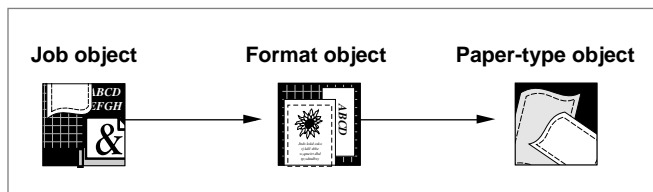
Core Printing Features

To enable these core features, your application must manipulate three kinds of objects:

- the **job object**, which contains information about the print job used to print a document
- the **format object**, which contains information about how to format one or more pages of a document for printing
- the **paper-type object**, which contains information about the paper on which a document is to be printed

Figure 2-1 shows the relationship between these objects.

Figure 2-1 Objects needed to implement core printing features



All aspects of printing with QuickDraw GX relate to a particular job object. The job object defines the parameters with which to print the document, which a user typically specifies in the Print dialog box.

Your application sets up the correspondence between a document and a job object. The job object is tied to the format and paper-type objects through references. A job object refers to at least one format object. The format object specifies how to format the pages in a document. To implement core printing features, in which each page of a document is formatted the same way, you are only concerned about the first reference to a format object because this format object represents the default format.

Each format object refers to a paper-type object. Thus, it is actually this pair of objects that specifies how the pages of a document are formatted. The user typically specifies the format options, which translate into format object properties and specifies paper-type options, which translate into paper-type object properties, in the Page Setup Dialog box.

These three objects—the job, format, and paper-type—can refer to other objects, some of which are collections of additional specifications. These other objects and specifications are not required, however, to implement the core printing features.

The references themselves are properties of the job, format, or paper-type objects. The references are mentioned in the following section, which describes each object's properties. The other objects themselves, however, are described as they are used in the chapters "Page Formatting and Dialog Box Customization" and "Advanced Printing Features" in this book.

Core Printing Features

In addition to manipulating job, format, and paper-type objects, your application must also initialize the printing environment, handle printing-related errors, and handle two situations that can arise when the user invokes a print dialog box:

- Your application must let QuickDraw GX know which Edit menu items are to be enabled when a QuickDraw GX dialog box is active. Although QuickDraw GX implements the Cut, Copy, Paste, and Clear menu items for you, you must specify which of these items are enabled. If other menu items are enabled, such as Undo, you must also handle the item as well as enable it.
- Your application must respond to printing event messages, which allows updating the screen when the user moves a dialog box. Printing event messages and movable dialog boxes are described in the chapter “Introduction to Printing With QuickDraw GX” in this book.

Your application should also handle printing from the Finder, which occurs when the user chooses Print from the Finder’s File menu or drags a document onto a desktop printer icon. Finally, your application can also handle printing of existing documents designed for printing with the Macintosh Printing Manager.

The following section, “Core Print Objects,” describes the QuickDraw GX objects needed to implement core printing features. The section “Using Core Printing Features” beginning on page 2-10 provides examples of code that implements these core features.

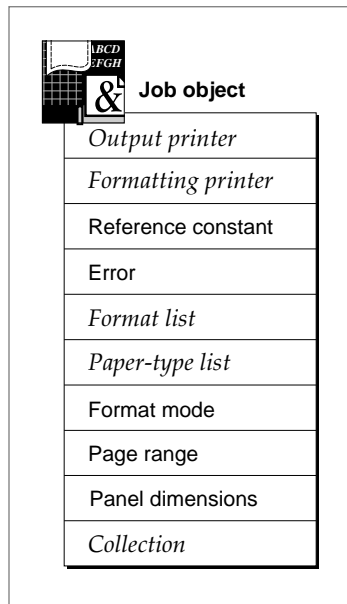
Core Print Objects

QuickDraw GX printing information is contained in a set of print objects that you associate with a printable document. When you create a job object, QuickDraw GX sets up references to a format object and paper-type object. The initial values for the properties in each of these objects is defined by the printer driver for the default output printer.

The following sections describe specific properties of the job object, the format object, and the paper-type object.

Job Object Properties

A job object has ten accessible properties, as shown in Figure 2-2. Note that, because the data structure of a job object is private, the order of the properties as shown in Figure 2-2 is completely arbitrary. Properties in *italics* indicate references to other objects.

Figure 2-2 The job object

The properties of a job object are as follows:

- **Output printer.** A reference to the output printer to which documents are sent for printing. A user specifies an output printer in the Print dialog box. The initial value contained in this property is the default output printer, which is the printer to which documents are sent if the user does not select a different printer.
- **Formatting printer.** A reference to the printer for which documents are formatted. A user specifies a formatting printer in the Page Setup dialog box. The initial value contained in this property is the default formatting printer, which is used to format documents if the user does not specify a different printer.
- **Reference constant.** This property contains a reference constant for your application's use. In the reference constant you can associate your own data with a particular job object. For example, you may wish to store a pointer to the document data. Specifying a reference constant for a job object is discussed in the chapter "Advanced Printing Features" in this book.
- **Error.** This property specifies the most recent error encountered for a particular job object. QuickDraw GX associates printing-related errors with individual job objects. It is necessary for you to check for errors after calling certain functions. Job object errors are discussed in "Error Handling" beginning on page 2-14.
- **Format mode.** This property specifies the mode associated with a particular job object. QuickDraw GX supports text, PostScript, and graphics direct modes. By default, your application uses the graphics direct mode to print text and graphics. Direct modes allow your application to take advantage of a printer's built-in features, such as fonts and text-streaming capabilities, to provide faster output for users. Using direct mode, however, does not take full advantage of QuickDraw GX features; therefore, the appearance of the document may change when printed in direct mode. The user can

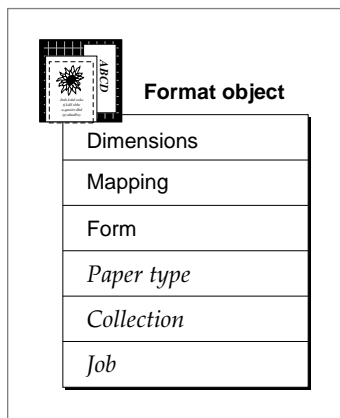
specify a direct mode in the Print dialog box. Direct modes are discussed in the chapter “Advanced Printing Features” in this book.

- **Format list.** A list of references to format objects. The first reference is to format object that represents the default format. The default format is defined by the printer driver of the default output printer. The user can change the value of the default format in the Page Setup dialog box. Using multiple format objects in a document is discussed in the chapter “Page Formatting and Dialog Box Customization” in this book. Format object properties are discussed in the next section.
- **Paper-type list.** A list of references to the paper-type objects that are associated with the job’s format objects. The user can change the default paper type in the Page Setup dialog box. Using different paper-type objects in a document is discussed in the chapter “Advanced Printing Features” in this book.
- **Page range.** This property contains the user-specified page range. A user specifies a page range in the Print dialog box. How you determine the page range is discussed in “Printing Documents Using QuickDraw GX” beginning on page 2-20.
- **Panel dimensions.** This property defines the dimensions of QuickDraw GX dialog box panels. You use this information when you want to locate the position of the cursor within a panel. Panel dimensions are discussed in the chapter “Page Formatting and Dialog Box Customization” in this book.
- **Collection.** A reference to a job collection object, which stores additional information about the print job. The job collection object is discussed in the chapter “Page Formatting and Dialog Box Customization” in this book.

Format Object Properties

A format object contains six accessible properties, as shown in Figure 2-3. Note that, because the data structure of a format object is private, the order of the properties as shown in Figure 2-3 is completely arbitrary. Properties in *italics* indicate references to other objects.

Figure 2-3 The format object



Core Printing Features

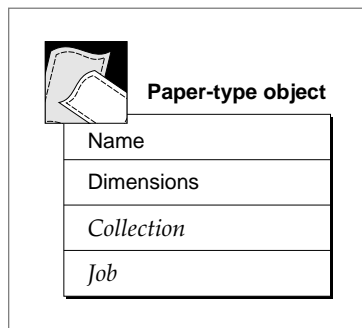
The properties of a format object are as follows:

- **Dimensions.** This property defines the physical dimensions of the paper (the paper size) and the printable area within these dimensions (the page size) after scaling and orientation have been applied. Scaling is the percentage that objects are shrunk or grown when printed. The orientation is either portrait or landscape.
- **Mapping.** This property defines the mathematical representation of the format object's settings, such as scaling. The mapping property of a format object is discussed in the chapter "Page Formatting and Dialog Box Customization" in this book.
- **Form.** This property defines a backdrop that can be applied to a set of pages. A **form** is made up of two shape objects—a shape that defines the form and another shape that defines a mask, which represents the erasable area within the form. Forms are discussed in the chapter "Page Formatting and Dialog Box Customization" in this book.
- **Paper type.** A reference to a paper-type object associated with this format object. Paper-type object properties are discussed in the next section.
- **Collection.** A reference to a format collection. Through this reference, you can access additional information related to the format collection. This information includes data such as the user-specified orientation (either portrait, landscape, or rotated landscape) from the Page Setup dialog box. The format collection is discussed in the chapter "Page Formatting and Dialog Box Customization" in this book.
- **Job.** A reference to a job object. Through this reference, you can access the job object associated with a particular format object.

Paper-Type Object Properties

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

Figure 2-4 The paper-type object



The properties of a paper-type object are as follows:

- **Name.** This property contains the name of a paper type, such as US Letter. A user specifies a paper-type name in the Page Setup or Custom Page Setup dialog box. Paper-type object names are discussed in the chapter “Advanced Printing Features” in this book.
- **Dimensions.** This property defines the physical dimensions of the paper (the paper size) and the printable area within these dimensions (the page size) before scaling and orientation have been applied. Paper-type object dimensions are discussed in the chapter “Advanced Printing Features” in this book.
- **Collection.** A reference to a paper-type collection. Through this reference, you can access additional information related to the paper-type object. This information includes such data as paper-type units. The paper-type collection is discussed in the chapter “Page Formatting and Dialog Box Customization” in this book.
- **Job.** A reference to a job object. Through this reference, you can access the job object associated with a particular paper-type object.

Edit Menu Structure

QuickDraw GX supports basic editing commands when a print dialog box is active. The user can Cut, Copy, Paste, and Clear edit text. To handle this task, QuickDraw GX must know the ID of the Edit menu, and the location within the edit menu of the items that correspond to Cut, Copy, Paste, and Clear.

Your application specifies this information in an Edit menu structure, named `gxEditMenuRecord`:

```
struct gxEditMenuRecord{
    short    editMenuID;
    short    cutItem;
    short    copyItem;
    short    pasteItem;
    short    clearItem;
    short    undoItem;
} ;
```

The `editMenuID` field specifies the ID of the Edit menu. The other fields identify the location of items in the Edit menu. For an example of how to set up an Edit menu structure, see “Displaying QuickDraw GX Print Dialog Boxes” beginning on page 2-35.

Note

QuickDraw GX does not support the Undo item. ♦

Because QuickDraw GX handles all menu items while a print dialog box is displayed, your application should disable all of its menus, except the Edit menu. It should also disable the About box under the Apple menu. Adjusting menus for movable modal dialog boxes such as print dialog boxes is described in *Inside Macintosh: Macintosh Toolbox Essentials*.

Using Core Printing Features

This section shows how to implement the core printing features in your application. First, you must determine if QuickDraw GX is installed and, if so, set up its environment. Next, when the user creates a document, your application needs to create a job object for the document and maintain other information about the document. The sample code throughout this book uses a structure, *MyDocumentRec*, to keep the needed information in one place:

```
typedef struct MyDocumentRec {
    gxJob      documentJob;      /* the job object bound to the
                                document */
    long       numPages;         /* the number of pages in the
                                document */
    long       curPage;          /* the current page */
    FSSpec     documentFSSpec;   /* the file system specification
                                for the document */
    Str31      documentTitle     /* the title of the document
                                (such as "Untitled") */
    WindowPtr  documentWindow;   /* the window for the document */
    gxViewPort documentViewPort; /* the view port used for
                                drawing within the document
                                window */
    gxShape     documentPage[kMaxPages]; /* the shape data for each
                                page */
    gxFormat    pageFormat[kMaxPages]; /* the format object for each
                                page, if nil use the default
                                format */
} MyDocumentRec, *MyDocumentPtr;
```

This structure is set up to handle one shape per page. Each page may have its own format, although in this chapter only one format is used. The individual fields in the structure are described as they are used in the following sections.

Your application could define a similar structure, or you could maintain the needed information in variables of your choosing. The variable used in this book is *myDocument*, which is defined as follows:

```
MyDocumentRec  myDocument;
```


Core Printing Features

The following sections show how to

- initialize QuickDraw GX printing
- create a job object and initialize the `myDocument` variable
- handle errors
- print a document
- save a job object by flattening it
- retrieve a job object by unflattening it
- dispose of a job object and the objects it references
- obtain format information
- support print dialog boxes
- perform printing from the Finder
- update job object information after resume events
- print existing documents designed for printing with the Macintosh Printing Manager

Initializing QuickDraw GX Printing

For your application to use QuickDraw GX, the user must be running system software version 7.1 or later. To test for the existence of QuickDraw GX printing features, use the `Gestalt` function. The Gestalt selector is `gestaltPrintingMgrVersion ('pmgr')`. The `Gestalt` function is discussed in *Inside Macintosh: QuickDraw GX Environment and Utilities*.

Note

The Gestalt selector for the entire QuickDraw GX feature set is `gestaltGXVersion`. This selector is discussed in *Inside Macintosh: QuickDraw GX Environment and Utilities*. ♦

After you call the `GXEnterGraphics` function to initialize QuickDraw GX, you call the `GXInitPrinting` function to initialize QuickDraw GX printing features. The `GXEnterGraphics` function is discussed in *Inside Macintosh: QuickDraw GX Environment and Utilities*.

Core Printing Features

To terminate printing with QuickDraw GX, you must call the `GXExitPrinting` function. You can only use this function after you have successfully called the `GXInitPrinting` function and before you call the `GXExitGraphics` function to shut down QuickDraw GX:

```
OSErr err;
...
GXEnterGraphics();
err = GXInitPrinting(); /* Set up print facility */
if (!err)
{
    /* The event loop and more initialization goes here */
    ...
}
GXExitPrinting();          /* Close QuickDraw GX printing. */
GXExitGraphics();
```

Creating a Job Object for a Printable Document

For each printable document that a user creates, your application needs to create a corresponding job object. Generally, you should manage job objects on a one-to-one basis with documents. An introduction to manipulating the job object in response to user actions is discussed in the chapter “Introduction to Printing With QuickDraw GX” in this book. Properties of the job object are described in “Job Object Properties” on page 2-5.

Listing 2-1 shows the `MyNewDocument1` function that creates a job object for a printable document and initializes a `MyDocumentRec` structure. The `docName` parameter of the `MyNewDocument1` function is a Pascal string containing the name of the document, and the `myDocument` parameter is a pointer to a `MyDocumentRec` structure. In this example, the document is simplified to handle a maximum of 20 pages.

Listing 2-1 Creating a job object for a printable document

```
#define kMaxPages    20

OSErr MyNewDocument1(Str31 docName, MyDocumentPtr myDocument)
{
    OSErr    err;
    Rect      bounds;

    myDocument->numPages = 0;          /* there are no pages yet */
    myDocument->curPage = 0;
```

Core Printing Features

```

/* Create a new job */
err = GXNewJob(&myDocument->documentJob);

if (err == noErr)
{
    /*
        Install your application override for the
        gxPrintingEvent message to display QuickDraw GX movable
        modal dialog boxes.
    */
    GXInstallApplicationOverride(myDocument->documentJob,
                                gxPrintingEvent,
                                MyPrintingEventOverride);

    /*
        Store the document's name. Limit is 31 characters (plus
        a length byte).
    */
    if (docName[0] > 31)
        docName[0] = 31;
    BlockMove(&docName[0], &myDocument->documentTitle[0],
              (long) docName[0] + 1);

    /*
        Additional application-specific document initialization
        can go here, such as the following:
        Create a window and a view port for the document. Store
        the pointer to the MyDocumentRec structure in the
        window's refCon field.
    */
    SetRect(&bounds, 30, 60, 300, 400);
    myDocument->documentWindow = NewCWindow(nil, &bounds,
                                             docName, false, noGrowDocProc, (WindowPtr) -1,
                                             true, (long) myDocument);
    err = MemError();
    if (err == noErr)
    {
        SetPort(myDocument->documentWindow);
        myDocument->documentViewPort =
            GXNewWindowViewPort(myDocument->documentWindow);
    }
}

```

Core Printing Features

```

        err = GXGetGraphicsError(nil);

        if (err != noErr)
            DisposeWindow(myDocument->documentWindow);
    }
    if (err != noErr) GXDisposeJob(myDocument->documentJob);
}
return err;
}

```

The `MyNewDocument1` function sets the number of pages in the document and the current page number. Note that pages begin at 1 (not from 0 as in an array). The initial value of 0 indicates that there are none.

The `GXNewJob` function creates a job object for the document. If an error does not occur, the `MyNewDocument1` function performs the following tasks:

- Calls the `GXInstallApplicationOverride` function to install a function that overrides the `gxPrintingEvent` message. This override is needed to handle movable print dialog boxes. The `GXInstallApplicationOverride` function and the `gxPrintingEvent` message are discussed in “Supporting QuickDraw GX Print Dialog Boxes” beginning on page 2-17.
- Stores the document’s name by calling the `BlockMove` function. The name is passed into the `MyNewDocument1` function.
- Creates the document’s window by calling the `NewWindow` function and makes it the focus by calling the `SetPort` function.
- Creates a view port for the window by calling the `GXNewWindowViewPort` function. This view port is used to draw individual shapes on a page and is discussed in the section “Printing Pages by Capturing Shapes” beginning on page 2-22. For information about the `GXNewWindowViewPort` function, see the environment chapter of *Inside Macintosh: QuickDraw GX Environment and Utilities*.

In the event of an error, the job and window are disposed of, if necessary.

Error Handling

QuickDraw GX provides you with an error-handling method to poll for printing-related errors. In previous versions of the Macintosh printing architecture, errors were handled using the `PrError` function. This function returned the error status. Printing errors were global to an application. In QuickDraw GX, an error is local to a job object.

You can poll for errors in two different ways: immediately after you call a function or after you call groups of functions. QuickDraw GX provides the `GXGetJobError` function to allow you to poll for errors in both ways.

When an error occurs, the error is stored in the error property of the job object. The error is not cleared until you call the `GXGetJobError` function. Thus, `GXGetJobError` returns the first error since the last call to the `GXGetJobError` function.

IMPORTANT

If an error condition exists for a job object, QuickDraw GX will not execute other functions associated with the job object until the error condition is cleared. ▲

You should note that it is necessary for you to check for errors after certain functions. For example, you should always check for errors after calling functions that begin with the word `Start` or functions related to collection objects. Functions related to collection objects are discussed in the chapter “Page Formatting and Dialog Box Customization” in this book.

Polling for errors is a standard Macintosh method used by the Resource Manager and the Macintosh Printing Manager. For information on the Resource Manager, see *Inside Macintosh: More Macintosh Toolbox*. For information on QuickDraw, see *Inside Macintosh: Imaging*.

Listing 2-2 shows an example of polling for errors by calling `GXGetJobError` after individual functions. The error conditions being checked for in the example can arise while executing the print loop. For a discussion of the print loop, see “Printing Documents Using QuickDraw GX” beginning on page 2-20.

Listing 2-2 Polling for errors after individual functions

```
GXGetJobPageRange(myDocument->documentJob, &firstPage, &lastPage);
err = GXGetJobError(myDocument->documentJob);
if (err == noErr)
{
    if (lastPage > myDocument->numPages)
        lastPage = myDocument->numPages;
    numPages = lastPage - firstPage + 1;

    GXStartJob(myDocument->documentJob,
               myDocument->documentTitle, numPages);
    err = GXGetJobError(myDocument->documentJob);
    if (err == noErr)
    {
        for (pg = firstPage; (err == noErr) && (pg <=
                                     lastPage); pg++)
        {
            GXPrintPage(myDocument->documentJob, pg,
                        GXGetJobFormat(myDocument->documentJob, 1),
                        myDocument->documentPage[pg - 1]);
            err = GXGetJobError(myDocument->documentJob);
        }
    }
}
```

Core Printing Features

```

        GXFinishJob(myDocument->documentJob);
        err = GXGetJobError(myDocument->documentJob);
    }
}

```

Listing 2-3 shows an example of polling for errors after groups of functions. This example shows how to obtain the dimensions of the paper and page associated with a format object, which is explained on page 2-33. If the `GXGetJobFormat` function returns an error, the `GXGetFormatDimensions` function returns immediately without executing.

Listing 2-3 Polling for errors after groups of functions

```

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

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

    /* Get the bounds of the format object. */
    GXGetFormatDimensions(pgFormat, pageBounds, paperBounds);

    return GXGetJobError(myDocument->documentJob);
}

```

Unless otherwise indicated, errors are generally checked after groups of functions throughout the code samples in this book.

Core Printing Features

In addition, QuickDraw GX allows you to store an error with a particular job object using the `GXSetJobError` function. This function is useful when you want to abort or cancel spooling, which is how data is sent to the printer driver. Spooling is discussed in the chapter “Introduction to Printing With QuickDraw GX” in this book.

The following statement sets the error condition associated with the job object to the contents of `err`:

```
GXSetJobError(myDocument->documentJob, err);
```

When the error status is tested using `GXGetJobError`, it will return the status set by the `GXSetJobError` function, assuming that another error did not occur between the time the value was set and then retrieved.

Supporting QuickDraw GX Print Dialog Boxes

Dialog boxes for QuickDraw GX printing features are movable modal. A movable modal dialog box is a modal dialog box that contains a title bar by which users can drag the dialog box. This type of dialog box allows users to view windows that would otherwise be obscured by the dialog box. Movable modal dialog boxes are described in *Inside Macintosh: Macintosh Toolbox Essentials*.

To support QuickDraw GX print dialog boxes, your application needs to identify the Edit menu and its menu items, adjust the menu bar to enable or disable appropriate menu items, and respond to the `gxPrintingEvent` message that QuickDraw GX sends to your application.

You make menu adjustments just before you display the dialog box. Examples of setting up the menu bar are shown in the sections “Displaying the Page Setup Dialog Box” beginning on page 2-35 and “Displaying the Print Dialog Box” beginning on page 2-37.

This section shows how to set up the override for the `gxPrintingEvent` message. QuickDraw GX sends this message to your application each time it receives an event, such as a mouse click or a keystroke. Because you want the application to respond to update events so that the window can be redrawn, you must install the application as a handler for the `gxPrintingEvent` message.

You create a function that has the same prototype (the same format of parameters and return value) as the `GXPrintingEvent` function and install it in the message chain. To override the `gxPrintingEvent` message, you specify a pointer to an override function in the `GXInstallApplicationOverride` function. Because dialog boxes are associated with individual job objects, you must call `GXInstallApplicationOverride` after you create each job object.

Core Printing Features

The override persists until you dispose of the job object or install another override for the `gxPrintingEvent` message. Listing 2-1 on page 2-12 shows the following call in the context of creating a new job object:

```
GXInstallApplicationOverride(myDocument->documentJob,
                             gxPrintingEvent,
                             MyPrintingEventOverride);
```

The `GXInstallApplicationOverride` function has three parameters:

- A reference to the job object. In Listing 2-1 on page 2-12, it is the job object that was stored when the document was created.
- The ID of the message to override. In this case, it is `gxPrintingEvent`.
- The function that responds to the message. In this case, it is `MyPrintingEventOverride`.

The parameters to the override function named `MyPrintingEventOverride` must match those of the `GXPrintingEvent` message override function, which has the following declaration:

```
OSErr GXPrintingEvent (EventRecord *anEventRecord,
                      Boolean filterEvent);
```

The `anEventRecord` parameter is a pointer to the event record, which contains information about what type of event occurred while the print dialog box was being displayed; for example, a mouse click or key-down. The event record also contains additional information associated with the event, such as which key was pressed for a key-down event.

The `filterEvent` parameter specifies whether the event can be filtered. QuickDraw GX sends two `gxPrintingEvent` messages for each event. The first event can be filtered, for example, by calling the `DialogSelect` function to filter non-update events.

Note

The Window Manager generates update events to control the appearance of windows on the screen. The `EventRecord` data type, the Window Manager, the `DialogSelect` function, and update events are discussed in *Inside Macintosh: Macintosh Toolbox Essentials*. ♦

Listing 2-4 shows an override function for the `gxPrintingEvent` message.

Listing 2-4 Override function for the `gxPrintingEvent` message

```
OSErr MyPrintingEventOverride(EventRecord *anEvent,
                              Boolean filterEvent)
{
    OSErr    err = noErr;

    /* Handle events in whatever way is appropriate. MyDoEvent
       is a generic event handler. Don't pass it events that
       it shouldn't handle while print dialogs are displayed.
    */
    if (!filterEvent)
        switch (anEvent->what)
        {
            case mouseDown:
            case keyDown:
            case autoKey:
                break;

            default:
                err = MyDoEvent(anEvent);
        }
    return err;
}
```

Note

You do not need to forward the `gxPrintingEvent` message. ♦

In Listing 2-4, if the event is not a filter event, `MyDoEvent` is called because the event is probably an update event. The `MyDoEvent` function is the general-purpose, application-specific function that handles all events and is typically called after each `WaitNextEvent`. The `MyDoEvent` function is called from this override to dispatch redrawing of the document's window in response to an update event.

Printing Documents Using QuickDraw GX

There are two approaches you can take to printing a document depending on how you store data. You can either print each page as a single picture shape or print each page by allowing QuickDraw GX to capture multiple shapes. In the later case, you specify when to start and stop capturing shapes that appear on the page.

If your application stores each page as a single picture shape, you should use the `GXPrintPage` function to print each page in a document. In the `GXPrintPage` function, you need to provide QuickDraw GX with the picture shape for each page. A picture shape is a container for other shapes—including other picture shapes, allowing you to create hierarchies of shapes. Picture shapes are discussed in *Inside Macintosh: QuickDraw GX Graphics*.

You may also choose to use the `GXStartPage`, `GXDrawShape`, and `GXFinishPage` functions to draw and print data. You should use these functions if your application does not store each page as a single picture shape. QuickDraw GX allows you to print in a way similar to how you draw to the screen except that QuickDraw GX captures shapes to send to a print file, such as a spool file or a portable digital document, instead of to a monitor. The `GXDrawShape` function is described in *Inside Macintosh: QuickDraw GX Objects*.

IMPORTANT

Some QuickDraw GX functions begin with the word `Start` or `Finish`. You must call the corresponding “finish” call only if the “start” call succeeds. For example, after you call the `GXStartPage` function, you should immediately check for errors. You should call the `GXFinishPage` function only if `GXStartPage` did not return an error. ▲

Regardless of whether you print pages as single picture shapes or print pages by capturing shapes, the basic flow of control is as follows:

- After the user requests printing, you call the `GXGetJobPageRange` function to obtain the user-specified page range.
- You use the `GXStartJob` function to begin printing a document with parameters that specify the job object and the name of the user’s document. You may also specify the total number of pages the user chose to print or pass 0 if the page count is unknown. In response to the `GXStartJob` call, QuickDraw GX displays the Status dialog box, which contains the current page number and the total page count, if it is not 0.
- After you finish printing, by either method, you call the `GXFinishJob` function to tell QuickDraw GX that the document is ready to be queued for printing in the background. Note that you should only call the `GXFinishJob` function if the `GXStartJob` function doesn’t return an error.

Printing Pages as Single Picture Shapes

This section describes how to use the `GXPrintPage` function to print a user's document. To use this function, you specify the page to print in the `pageNumber` parameter. QuickDraw GX compares the specified page number with the page range chosen by the user and spools the page if it is within the page range. If it is not within range, the page is ignored.

You should loop through each page of a document, calling the `GXPrintPage` function for each page's picture shape. You should check for errors after you print each page and exit the loop if an error arises.

Listing 2-5 gives an example of how to use the `GXPrintPage` function to print a document. In the example, only the default format is used to format each page. To obtain this format, you call the `GXGetJobFormat` function with an index of 1.

Listing 2-5 Using the `GXPrintPage` function to print a document

```
OSErr MyPrintDocument2(MyDocumentPtr myDocument)
{
    OSErr err;
    long firstPage, lastPage, numPages, pg;

    /* Determine which pages the user selected to print. */
    GXGetJobPageRange(myDocument->documentJob,
                      &firstPage, &lastPage);

    if (lastPage > myDocument->numPages)
        lastPage = myDocument->numPages;

    /*
       Calculate the total number of pages to print. If there are
       no errors, begin printing.
    */
    numPages = lastPage - firstPage + 1;
    err = GXGetJobError(myDocument->documentJob);
    if (err == noErr)
    {
        GXStartJob(myDocument->documentJob,
                   myDocument->documentTitle, numPages);
        err = GXGetJobError(myDocument->documentJob);
    }
}
```

Core Printing Features

```

/*
    Loop through each page. Call the GXPrintPage function for
    each page's picture shape. In this example, we use the
    job's default format to print each page.
*/
if (err == noErr)
{
    for (pg = firstPage; (err == noErr) && (pg <= lastPage);
        pg++)
    {
        GXPrintPage(myDocument->documentJob, pg,
                    GXGetJobFormat(myDocument->documentJob, 1),
                    myDocument->documentPage[pg - 1]);
        err = GXGetJobError(myDocument->documentJob);
    }

    /* Finish printing. */
    if (err == noErr)
    {
        GXFinishJob(myDocument->documentJob);
        err = GXGetJobError(myDocument->documentJob);
    }
}
return err;
}

```

Printing Pages by Capturing Shapes

This section describes how to use the `GXStartPage`, `GXDrawShape`, and `GXFinishPage` functions to print pages in your application's documents. You use the `GXStartPage` function to tell QuickDraw GX to capture shapes that you draw using the `GXDrawShape` function. You call `GXFinishPage` when you are finished creating the page of output.

In the `GXStartPage` function, you set the page to print in the `pageNumber` parameter. QuickDraw GX compares the specified page number with the page range chosen by the user and spools the page if it is within the page range. If it is not within range, the page is ignored.

Core Printing Features

In the `GXStartPage` function, you also specify a `viewPortList` parameter, which is the list of view ports to use to capture shapes. The part of the shape that can be drawn through the view port is spooled. In the `numViewPorts` parameter, you specify the number of view ports to use (as specified in the `viewPortList` parameter). QuickDraw GX drawing functions and view port objects are described in *Inside Macintosh: QuickDraw GX Objects*.

Note

QuickDraw GX does not use the information in a view port, such as its mapping or clipping properties. It uses a view port only to capture the shape information, such as the geometry and color, as shapes are drawn. For example, you can print as you draw by specifying view ports in the onscreen view group in the call to `GXStartPage`, or you can draw to offscreen view ports to capture shapes without displaying them. In either case, only the information about the shape is spooled. ♦

Listing 2-6 gives an example of how to print a document using the `GXStartPage`, `GXDrawShape`, and `GXFinishPage` functions.

Listing 2-6 Using the `GXStartPage`, `GXDrawShape`, and `GXFinishPage` functions to print a document

```
OSErr MyPrintDocument2(MyDocumentPtr myDocument)
{
    OSErr    err;
    long     firstPage, lastPage, numPages, pg;

    /* Determine which pages the user selected to print. */
    GXGetJobPageRange(myDocument->documentJob, &firstPage,
                     &lastPage);
    if (lastPage > myDocument->numPages)
        lastPage = myDocument->numPages;

    /* Calculate the total number of pages to print. */
    numPages = lastPage - firstPage + 1;
    err = GXGetJobError(myDocument->documentJob);

    /* Begin printing if there are no errors. */
    if (err == noErr)
    {
        GXStartJob(myDocument->documentJob,
                  myDocument->documentTitle, numPages);
    }
}
```

Core Printing Features

```

/*
    For each page, call the GXStartPage function, draw the
    page, and then call the GXFinishPage function. In this
    example, the default format and the document's view
    port are used, drawing only a single shape on each page.
*/
for (pg = firstPage; (err == noErr) && (pg <= lastPage);
    pg++)
{
    /* Start the page. */
    GXStartPage(myDocument->documentJob, pg,
                GXGetJobFormat(myDocument->documentJob, 1),
                1, &myDocument->documentViewPort);
    err = GXGetJobError(myDocument->documentJob);

    /* If there are no errors, draw the data for the page. */
    if (err == noErr)
    {
        GXDrawShape(myDocument->documentPage[pg - 1]);
        err = (OSErr) GXGetGraphicsError(nil);
    }
    if (err == noErr)
        GXFinishPage(myDocument->documentJob);
}

/* Finish printing. */
GXFinishJob(myDocument->documentJob);
err = GXGetJobError(myDocument->documentJob);
}
return err;
}

```

Saving a Job Object With a Document File

There are two approaches you can take to saving a job object with its corresponding document. Either you can create a handle in which to store the job object and then flatten the job object into this handle, or you can specify a pointer to a flattening function to flatten the job object and save its data to disk.

When the user chooses the Save or Save As menu command from the File menu, you should save the document and its corresponding job object to disk. To save a job object, you flatten it. To retrieve a job object, you unflatten it. For an introduction to flattening and unflattening QuickDraw GX print objects, see the chapter “Introduction to Printing With QuickDraw GX” in this book.

Core Printing Features

When a user saves a document, you may prefer to save a job object in a single handle using the `GXFlattenJobToHdl` function. You may also choose to use the `GXFlattenJob` function to save a job object. You specify a pointer to a flattening function in this function because it requires less memory to save portions of job object data to disk than it does to save the data in a single handle.

Saving a Job Object in a Single Handle

This section describes how to use the `GXFlattenJobToHdl` function to save a job object and its related data.

You should create a handle in which to store the job object and then flatten the job object into this handle. You specify a handle in the `aHandle` parameter to the `GXFlattenJobToHdl` function. QuickDraw GX grows or shrinks the size of the handle you provide to accommodate the size of the job object. You then save the contents of the handle, typically in the document's resource fork.

Listing 2-7 shows how to save a job object in a document using the `GXFlattenJobToHdl` function.

Listing 2-7 Using the `GXFlattenJobToHdl` function to save a job object

```
OSErr MySaveDocument(MyDocumentPtr myDocument)
{
    OSErr      err;
    Handle     theJobData, oldJobData;
    short      dataRefNum = -1;
    short      oldResFile, resRefNum = -1;
    FSSpec     *docFSSpec;

    /*
       Create a handle in which to store the job object and then
       flatten the job object into this handle.
    */
    oldResFile = CurResFile();
    theJobData = NewHandle(0);
    err = MemError();
    if (err == noErr)
    {
        GXFlattenJobToHdl(myDocument->documentJob, theJobData);
        err = GXGetJobError(myDocument->documentJob);

        if (err == noErr)
        {
```

Core Printing Features

```

/* Open the file's data fork and resource fork. */
docFSSpec = &myDocument->documentFSSpec;
err = FSpOpenDF(docFSSpec, fsRdWrPerm, &dataRefNum);
if (err == noErr)
{
    resRefNum = HOpenResFile(docFSSpec->vRefNum,
                             docFSSpec->parID,
                             docFSSpec->name, fsRdWrPerm);

    err = ResError();
}

/* Delete any existing job object resources. */
if (err == noErr)
{
    UseResFile(resRefNum);
    oldJobData = Get1Resource(kMyJobType, kMyJobID);
    if (oldJobData != nil)
    {
        RmveResource(oldJobData);
        UpdateResFile(resRefNum);
        DisposHandle(oldJobData);
    }

    /* Add the new job object resource. */
    AddResource(theJobData, kMyJobType, kMyJobID, "\p");
    err = ResError();
    if (err == noErr)
    {
        WriteResource(theJobData);
        UpdateResFile(resRefNum);
        ReleaseResource(theJobData);
    }
}

/*
Write the data for a document's pages to the data
fork. Place your application-specific code here to
save page data associated with the document.
*/
...
}

```


Core Printing Features

```

        /* Close the data and resource forks of this document. */
        if (dataRefNum != -1) FSClose(dataRefNum);
        if (resRefNum != -1) CloseResFile(resRefNum);
    }
    else
        DisposHandle(theJobData);
}
UseResFile(oldResFile);
return err;
}

```

Saving a Job Object Using a Flattening Function

This section describes how to use the `GXFlattenJob` function to save a job object. You specify a pointer to a flattening function in the `aPrintingFlattenProc` parameter of this function.

An example of a flattening function named `MyFlattenFunction` that you could write is as follows:

```

OSErr MyFlattenJobFunc(long dataSize, void *data,
                      void *dataRefNum)
{
    long count = dataSize;
    return FSWrite((short) dataRefNum, &count, data);
}

```

QuickDraw GX calls your flattening function multiple times as it saves job object data to disk. The `dataSize` parameter specifies the number of bytes for this segment of the job object data. The `data` parameter specifies a pointer to the segments of job object data to write out. The `dataRefNum` parameter specifies the file reference number of the open file to which you want to write.

Core Printing Features

Listing 2-8 shows how to save a job object using the `GXFlattenJob` function.

Listing 2-8 Using the `GXFlattenJob` function to save a job object

```
OSErr MySaveJobInDataFork(MyDocumentPtr myDocument,
                          short dataRefNum)
{
    OSErr    err;

    /*
     * Reset the file's position to the beginning of the data fork
     * and write the flattened job object there.
     */
    err = SetFPos(dataRefNum, fsFromStart, 0);
    if (err == noErr)
    {
        GXFlattenJob(myDocument->documentJob,
                     (gxPrintingFlattenProc) MyFlattenJobFunc,
                     dataRefNum);
        err = GXGetJobError(myDocument->documentJob);
    }
    return err;
}
```

Disposing of a Job Object When Closing a Document

When the user chooses the Close menu command from the File menu to close a document, you need to dispose of its job object. You should not dispose of a job object while its document is open.

For each page in a document, you should dispose of the page's shape. You can then call the `GXDisposeJob` function to dispose of a document's job object and associated format objects. Listing 2-9 shows how to dispose of a job object when a user closes a document.

Listing 2-9 Disposing of a job object when you close a document

```

OSErr MyCloseDocument(MyDocumentPtr myDocument)
{
    OSErr    err = noErr, jobErr;
    long     pg;

    /* Dispose of each page's shape */
    for (pg = 1; pg <= myDocument->numPages; pg++)
        GXDisposeShape(myDocument->documentPage[pg-1]);

    /* Dispose of the document's corresponding job object. */
    err = GXDisposeJob(myDocument->documentJob);

    /*
       Place any application-specific code here to close a
       document.
    */
    ...
    DisposeWindow(myDocument->documentWindow);
    return err;
}

```

Note

The `GXDisposeJob` function returns an error because errors are job-oriented. You cannot query a job object for errors once you have disposed of it. ♦

Retrieving a Job Object When Opening a Document

When the user chooses the Open menu command from the File menu to open a document, you need to retrieve its job object. To retrieve a job object, you unflatten it using one of the QuickDraw GX unflattening functions. For an introduction to flattening and unflattening QuickDraw GX print objects, see the chapter “Introduction to Printing With QuickDraw GX” in this book.

There are two methods to retrieving a job object depending on how you have previously saved it. If you saved the job object using the `GXFlattenJobToHdl` function, you should retrieve it using the `GXUnflattenJobFromHdl` function. If you saved the job object using the `GXFlattenJob` function, you should retrieve it using the `GXUnflattenJob` function. For details on the `GXFlattenJobToHdl` and `GXFlattenJob` functions, see “Saving a Job Object With a Document File,” which begins on page 2-24.

Retrieving a Job Object From a Handle

This section describes how to use the `GXUnflattenJobFromHdl` function to retrieve a job object and its related data.

When a user chooses the Open menu command from the File menu, you should open the document and retrieve its previously saved job object. To do so, you open the document's data fork and resource fork. The `MyOpenDocument` function in Listing 2-10 accomplishes this.

If there are no errors, you should specify the document's file system specification information, its title, and its window's title. If there is a job object resource saved in the resource file, you should load it and unflatten it using the `GXUnflattenJobFromHdl` function.

After the job object is unflattened, you can load the data for the document's pages. Finally, you should close the document's data fork and resource fork. Listing 2-10 shows how to open a document and retrieve its job object using the `GXUnflattenJobFromHdl` function.

Listing 2-10 Using the `GXUnflattenJobFromHdl` function to retrieve a job object

```
OSErr MyOpenDocument(MyDocumentPtr myDocument)
{
    OSErr          err;
    Handle          theJobData;
    short           oldResFile;
    short           dataRefNum = -1, resRefNum = -1;
    StandardFileReply sfReply;
    SFTYPEList      myTypeList;

    /* Let the user select a document to open. */
    oldResFile = CurResFile();

    myTypeList[0] = kMyDocType;
    StandardGetFile(nil, 1, &myTypeList, &sfReply);
    if (!sfReply.sfGood)
        return noErr;

    /* Open the selected file's data fork and resource fork. */
    err = FSpOpenDF(&sfReply.sfFile, fsRdWrPerm, &dataRefNum);
    if (err == noErr)
    {
        resRefNum = HOpenResFile(sfReply.sfFile.vRefNum,
                                sfReply.sfFile.parID,
                                sfReply.sfFile.name, fsRdPerm);
```

Core Printing Features

```

        err = ResError();
    }
    if (err) return err;

    /*
     * If no error, set the document's file system specification
     * information, its title, and its window's title.
     */
    BlockMove(&sfReply.sfFile, &myDocument->documentFSSpec,
              sizeof(FSSpec));
    BlockMove(&sfReply.sfFile.name, myDocument->documentTitle,
              (long) sfReply.sfFile.name[0] + 1);
    SetWTitle(myDocument->documentWindow,
              myDocument->documentTitle);

    /*
     * If there's a job object resource saved,
     * load and unflatten it.
     */
    UseResFile(resRefNum);
    theJobData = Get1Resource(kMyJobType, kMyJobID);
    if (theJobData != nil)
    {
        GXUnflattenJobFromHdl(myDocument->documentJob, theJobData);
        err = GXGetJobError(myDocument->documentJob);
        ReleaseResource(theJobData);
    }

    /*
     * Place your application-specific code here to load
     * other data associated with the document's pages.
     */
    ...

    /* Close the data fork and resource fork of this document. */
    if (dataRefNum != -1) FSClose(dataRefNum);
    if (resRefNum != -1) CloseResFile(resRefNum);
    UseResFile(oldResFile);
    return err;
}

```

Retrieving a Job Object Using an Unflattening Function

This section describes how to use the `GXUnflattenJob` function to retrieve a job object. You specify a pointer to an unflattening function in the `aPrintingFlattenProc` parameter of the `GXUnflattenJob` function.

An example of an unflattening function named `MyUnflattenFunction` that you could write is as follows:

```
OSErr MyUnflattenJobFunc(long dataSize, void *data,
                        void *dataRefNum)
{
    long count = dataSize;
    return FSRead((short) dataRefNum, &count, data);
}
```

QuickDraw GX calls your unflattening function multiple times as it retrieves job object-related data from disk. The `dataSize` parameter specifies the number of bytes for this segment of the job object data. The `data` parameter specifies a pointer to the segments of job object data to read. The `dataRefNum` parameter specifies the file reference number of the open file from which you want to read.

Listing 2-11 shows how to retrieve a job object using the `GXUnflattenJob` function.

Listing 2-11 Using the `GXUnflattenJob` function to retrieve a job object

```
OSErr MyLoadJobFromDataFork(MyDocumentPtr myDocument,
                          short dataRefNum)
{
    OSErr err;

    /*
     * Reset the file's position to the beginning of the data fork,
     * read and then unflatten the job object from there.
     */
    err = SetFPos(dataRefNum, fsFromStart, 0);
    if (err == noErr)
    {
        GXUnflattenJob(myDocument->documentJob,
                      (gxPrintingFlattenProc) MyUnflattenJobFunc,
                      (void *) dataRefNum);
        err = GXGetJobError(myDocument->documentJob);
    }
    return err;
}
```

Obtaining Object References

A job object can reference several format objects. Once you know which format object you want, you can access its properties. QuickDraw GX provides the `GXGetFormatJob` function to determine which job object is associated with a particular format object. Even if you know the format's job, you may still want to examine all references to the job's format objects. You can obtain these references with the `GXGetJobFormat` function.

Listing 2-12 shows an example that uses the `GXGetFormatJob` function to obtain the job object that references a format object and then loops through all the job's format objects using the `GXGetJobFormat` function. The example's function, `MyGetFormatIndex`, returns the format's position, or index value, of the specified format object in the job's list of format objects.

Listing 2-12 Using the `GXGetFormatJob` function to obtain a job object

```
long MyGetFormatIndex(gxFormat myFormat)
{
    gxJob    formatsJob;
    long     idx, numFormats;

    /*
     * Obtain the job object and count of the number of format objects
     * it references.
     */
    formatsJob = GXGetFormatJob(myFormat);
    numFormats = GXCountJobFormats(formatsJob);

    /*
     * Compare each of the references to locate the specified format
     * object and return the current format object index.
     */
    for (idx = 1; idx <= numFormats; ++idx)
        if (myFormat == GXGetJobFormat(formatsJob, idx))
            return idx;
}
```

Obtaining Information From a Format Object

This section provides an example of how to obtain information from a format object. QuickDraw GX provides functions that allow you to get, and in some cases set, the values of printing-related object properties.

This example uses the `GXGetFormatDimensions` function, which returns the dimensions property of a format object. The dimensions property includes the physical dimensions of the paper (the paper size) and the printable area within these

Core Printing Features

dimensions (the page size) after scaling and orientation have been applied. For a discussion of how the dimensions can be scaled or otherwise changed, see the chapter “Page Formatting and Dialog Box Customization” in this book.

Listing 2-13 shows how to use the `GXGetFormatDimensions` function to obtain a format object’s dimensions property.

Listing 2-13 Using the `GXGetFormatDimensions` function

```
OSErr MyGetFormatDimensions(MyDocumentPtr myDocument,
                           gxRectangle *pageBounds,
                           gxRectangle *paperBounds)
{
    long          curPage;
    gxFormat      pgFormat;

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

    /* Get the bounds of the format object. */
    GXGetFormatDimensions(pgFormat, pageBounds, paperBounds);

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

Note

The `GXGetFormatDimensions` function returns both the page size and the paper size of a particular document. Most applications are generally interested in only the page size, so QuickDraw GX allows you to pass `nil` for the pointer to the paper size. ♦

Displaying QuickDraw GX Print Dialog Boxes

You call functions to display most QuickDraw GX print dialog boxes. You use the `GXJobDefaultFormatDialog` function to display the Page Setup dialog box, and you use the `GXJobPrintDialog` function to display the Print dialog box. You use the `GXFormatDialog` function to display the Custom Page Setup dialog box, which is discussed in the chapter “Page Formatting and Dialog Box Customization” in this book.

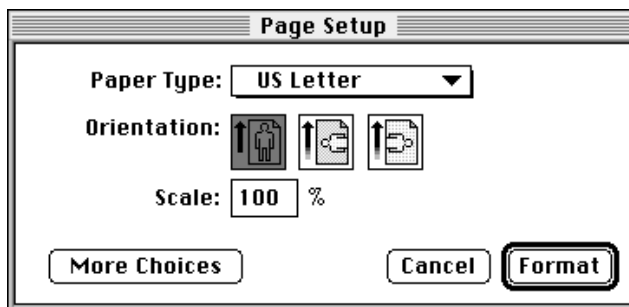
Displaying the Page Setup Dialog Box

When the user chooses the Page Setup menu command from the File menu, you call the `GXJobDefaultFormatDialog` function to display the Page Setup dialog box. In this dialog box, the user can specify formatting information for the default format. For example, the user can specify the paper type, orientation, and scaling.

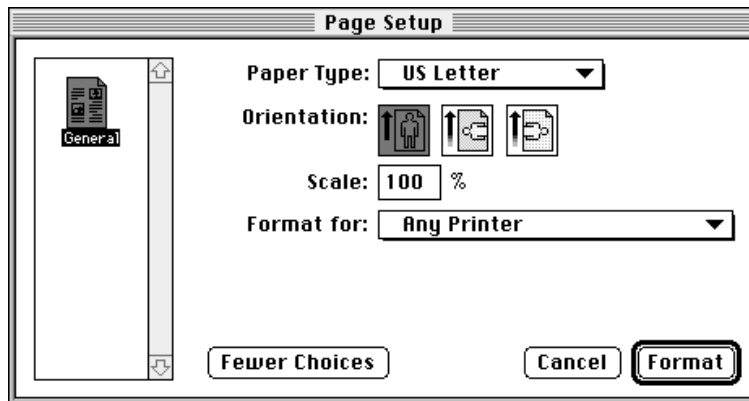
QuickDraw GX stores a user’s responses to some dialog items in the Page Setup dialog box in a format collection. QuickDraw GX stores default items, such as these, for you automatically. The format collection is discussed in the chapter “Page Formatting and Dialog Box Customization” in this book.

Figure 2-5 shows the Page Setup dialog box the user sees when you call the `GXJobDefaultFormatDialog` function.

Figure 2-5 The Page Setup dialog box



If the user chooses More Choices in the Page Setup dialog box, QuickDraw GX expands the dialog box. Figure 2-6 shows the expanded Page Setup dialog box. The expanded dialog box in this figure only contains one panel, the General panel. A printer driver, printing extension, or application can customize the dialog box to add additional panels. For more information about adding panels, see the chapter “Page Formatting and Dialog Box Customization” in this book.

Figure 2-6 The expanded Page Setup dialog box

Listing 2-14 shows the `MyFormatDialog` function, which calls the `GXJobDefaultFormatDialog` function to display the Page Setup dialog box. The Edit menu structure, `gxEditMenuRecord`, is set up before the dialog box is displayed. For information about the Edit menu structure, see “Edit Menu Structure” beginning on page 2-9. If the user chooses the Format button and there are no errors, document formatting can proceed.

Listing 2-14 Displaying the Page Setup dialog box

```
#define mEdit      128

#define kUndo      1
#define kCut       3
#define kCopy      4
#define kPaste     5
#define kClear     6
...
OSErr MyFormatDialog(MyDocumentPtr myDocument)
{
    OSErr          err;
    gxDialogResult  result;
    gxEditMenuRecord editMenuRec;

    /* Fill in the location of your application's Edit menu items. */

    editMenuRec.editMenuID = mEdit;
    editMenuRec.cutItem    = kCut;
    editMenuRec.copyItem   = kCopy;
    editMenuRec.pasteItem   = kPaste;
```

Core Printing Features

```

        editMenuRec.clearItem    = kClear;
        editMenuRec.undoItem    = kUndo;

/* Display the Page Setup dialog box. */
    result = GXJobDefaultFormatDialog(myDocument->documentJob,
                                      &editMenuRec);
    err = GXGetJobError(myDocument->documentJob);

/*
    If the user chooses the Format button and there are no
    errors, perform document formatting.
*/
    if ((err == noErr) && (result == gxOKSelected))
    {
        /*
            Place your application-specific code here if you need
            to repaginate the document.
        */
        ...
    }

    return err;
}

```

Displaying the Print Dialog Box

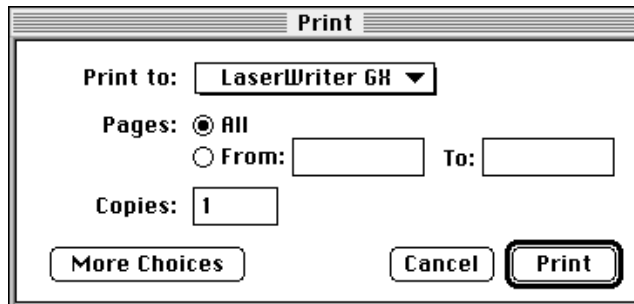
When the user chooses the Print menu command from the File menu, you call the `GXJobPrintDialog` function to display the Print dialog box. In this dialog box, the user can specify information related to actual printing of the document. For example, in the panels of the Print dialog box the user can specify the printer, print quality, number of copies to print, page range, automatic or manual paper feed, and whether a document should be sent to a printer or a file.

QuickDraw GX stores a user's responses to some dialog items in the Print dialog box in a job collection. The job collection is discussed in the chapter "Page Formatting and Dialog Box Customization" in this book.

Core Printing Features

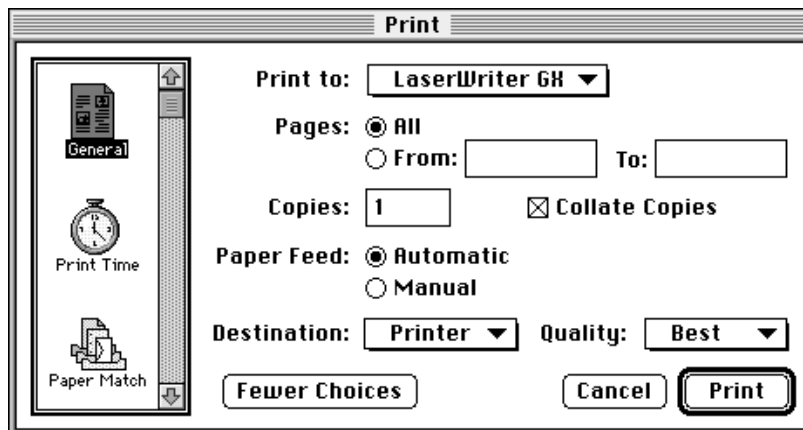
Figure 2-7 shows the Print dialog box the user sees when you call the `GXJobPrintDialog` function.

Figure 2-7 The Print dialog box



If the user chooses More Choices in the Print dialog box, QuickDraw GX expands the dialog box. Figure 2-8 shows the expanded Print dialog box. The expanded dialog box includes the standard panels (General, Print Time, and Paper Match), and any panels added by the application, printing extensions, or a printer driver.

Figure 2-8 The expanded Print dialog box



Listing 2-15 shows the `MyPrintDialog` function, which calls the `GXJobPrintDialog` function to display the Print dialog box. The Edit menu structure, `gxEditMenuRecord`, is set up before the dialog box is displayed. For information about the Edit menu structure, see “Edit Menu Structure” beginning on page 2-9. If the user chooses the Print button and there are no errors, printing can proceed.

Listing 2-15 Displaying the Print dialog box

```

OSErr MyPrintDialog(MyDocumentPtr myDocument)
{
    OSErr          err;
    gxDialogResult  result;
    gxEditMenuRecord editMenuRec;

    /* Fill in the location of your application's Edit menu items. */

    editMenuRec.editMenuID = mEdit;
    editMenuRec.cutItem    = kCut;
    editMenuRec.copyItem   = kCopy;
    editMenuRec.pasteItem  = kPaste;
    editMenuRec.clearItem  = kClear;
    editMenuRec.undoItem   = kUndo;

    /* Display the Print dialog box. */
    result = GXJobPrintDialog(myDocument->documentJob,
                             &editMenuRec);
    err = GXGetJobError(myDocument->documentJob);

    /*
     * If the user chooses the Print button and there are no errors,
     * call your printing function to print the pages.
     */
    if ((err == noErr) && (result == gxOKSelected))
        err = MyPrintDocument(myDocument);

    return err;
}

```

Supporting Printing From the Finder

A user can print from the Finder in two ways. A user can select a document and then choose the Print menu command from the File menu, or the user can drag a document to a desktop printer icon. To support printing from the Finder, your application must respond to the Print Documents ('pdoc') Apple event. Apple events provide your application with a standard mechanism for communicating with other applications.

To handle the Print Documents event, your application should print the documents specified in the Apple event. You can determine whether a document was dragged to a desktop printer icon by checking the `keyOptionalKeywordAttr` attribute of the Print Documents Apple event. Your application extracts this information and then prints the specified documents. Your application should not open any windows for the documents.

Core Printing Features

The Print Documents Apple event is discussed in the Apple events chapter of *Inside Macintosh: Interapplication Communication*.

Your application is responsible for determining the output printer on which to print the document. When a user drags a document to a desktop printer icon, your application must call the `GXSelectJobOutputPrinter` function to specify the output printer on which to print the selected document. This call is necessary because the document may have been printed previously and that job information may have been saved with the document. The `GXSelectJobOutputPrinter` function allows you to reselect the printer.

Listing 2-16 shows how to respond to the Print Documents Apple event and specify an output printer.

Listing 2-16 Responding to the Print Documents Apple event and specifying an output printer

```
pascal OSErr MyHandlePDOC(AppleEvent *theAppleEvent,
                          AppleEvent *reply, long myRefCon)
{
    OSErr          err;
    AEDescList      docList, dtpList;
    FSSpec          myFSS, dtpFSS;
    long            itemsInList, i;
    AEKeyword       theKeyword;
    DescType        typeCode;
    Boolean          draggedToDTP = false;
    Size            actualSize;
    MyDocumentRec   myDocument;

    /* Get the document list. */
    err = AEGetParamDesc(theAppleEvent, keyDirectObject,
                        typeAEList, &docList);
    if (err) return err;

    /*
     * Check to see if the user dragged the document to a desktop
     * printer.
     */
    err = AEGetParamDesc(theAppleEvent, keyOptionalKeywordAttr,
                        typeAEList, &dtpList);
    if (err == noErr) draggedToDTP = true;
```

Core Printing Features

```

/*
    Make sure you've accounted for all of the parameters passed
    and count the number of documents specified.
*/
err = MyCheckAEPParams(theAppleEvent);
if (err) return err;
err = AECCountItems(&docList, &itemsInList);
if (err) return err;

/*
    If the user dragged the document to a desktop printer, get the
    name of the desktop printer and throw away its description
    list.
*/
if (draggedToDTP)
{
    err = AEGGetNthPtr(&dtpList, 1, typeFSS, &theKeyword,
                      &typeCode, (Ptr) &dtpFSS,
                      sizeof(FSSpec), &actualSize);
    AEDisposeDesc(&dtpList);
}

/*
    For each entry in the document list, load it, print it, and
    close it.
*/
for (i = 1; i <= itemsInList; err == noErr; i++)
{
    err = AEGGetNthPtr(&docList, i, typeFSS, &theKeyword,
                      &typeCode, (Ptr) &myFSS, sizeof(FSSpec),
                      &actualSize);

    if (err == noErr)
    {
        /* Load the document. */
        err = MyNewDocument("\p", &myDocument);
        if (err == noErr)
        {
            err = MyFSOpenDocument(&myDocument, &myFSS);
            if (err == noErr)

```

Core Printing Features

```

        /*
        If the user dragged the document to a desktop
        printer, select this printer as the output printer
        for each job object.
        */
        {
            if (draggedToDTP)
                GXSelectJobOutputPrinter(myDocument.documentJob,
                                         dtpFSS.name);

            err = MyPrintDocument(&myDocument);
        }

        /* Close the document once it's printed. */
        MyCloseDocument(&myDocument);
    }
}

/* When you're done, throw away the document list. */
AEDisposeDesc(&docList);
return err;
}

```

Updating Job Object Information

When you receive a resume event, you should use the `GXUpdateJob` function to update the job object because the printing environment may have changed while the user was switched out of your application. For example, the user may have changed the desktop printer's settings, such as paper-tray information, while using another application.

Listing 2-17 shows an example of how to update the job object for a document. The `GXUpdateJob` function is called from the `MyDoEvent` function in response to a resume event.

Listing 2-17 Updating a job when receiving resume events

```

OSErr MyDoEvent(EventRecord *event)
{
    OSErr          err = noErr;
    WindowPtr      curWindow;
    MyDocumentPtr  windowDoc;

    switch (event->what)
    {
        /*
         * Application-specific code to handle mouse-down events,
         * update events, and so on.
         */
        case osEvt:
            switch ((event->message >> 24) & 0x0FF)
            {

                case suspendResumeMessage:
                    SetCursor(&qd.arrow);

                /* On a suspend event, coerce the scrap. */
                if ((event->message & resumeFlag) == 0)
                {
                    ZeroScrap();
                    TEToScrap();
                }
                else
                {

                    /*
                     * On a resume event, call GXUpdateJob on all of the documents'
                     * job objects. The user may have just changed something which
                     * affects the job objects, such as the size of the paper in the
                     * printer.

```

Core Printing Features

Since your application stores the document pointers in the reference constant fields of the documents' windows, loop through each window, extract the document pointers, and update the associated job objects.

```

*/
        if (event->message & convertClipboardFlag)
            TEFFromScrap();
        curWindow = FrontWindow();
        while (curWindow != nil)
        {
            if (((WindowPeek) curWindow)->windowKind ==
                userKind)
            {
                windowDoc = (MyDocumentPtr)
                    GetWRefCon(curWindow);
                GXUpdateJob(windowDoc->documentJob);
            }
            curWindow = (WindowPtr) (((WindowPeek)
                curWindow)->nextWindow);
        }
        break;
    }
    break;

    /*
        Application-specific code to handle high-level events.
    */
}
return err;
}

```

Printing Macintosh Printing Manager Documents

Documents printed with applications that use the Macintosh Printing Manager can be printed on a system with QuickDraw GX installed without the application being aware that QuickDraw GX is installed. Printing in this way, however, does not allow the application to take advantage of QuickDraw GX printing features, such as additional options provided in QuickDraw GX print dialog boxes, formatting, customization, and so on.

Core Printing Features

You can modify an existing application to allow it to print a document designed for printing with the Macintosh Printing Manager by determining whether QuickDraw GX is installed and, if it is, performing these steps:

1. Convert the print record associated with a Macintosh Printing Manager document into a job object.
2. Install the QuickDraw GX Translator to convert the results of QuickDraw functions into special QuickDraw GX shape objects that are used to spool QuickDraw output.
3. Execute your print loop. See the section “Printing Documents Using QuickDraw GX” beginning on page 2-20 for an example.
4. Remove the QuickDraw GX Translator.

To convert the print record associated with a Macintosh Printing Manager document into a job object, use the `GXConvertPrintRecord` function. Listing 2-18 shows how to use the `GXConvertPrintRecord` function.

Listing 2-18 Converting a print record into a job object

```
OSErr MyPrintRecordToJob(MyDocumentPtr myDocument, THPrint hPrint)
{
    /*
     * Convert the print record and store its settings in
     * the specified job object. Dispose of its handle.
     */
    GXConvertPrintRecord(myDocument->documentJob, hPrint);
    DisposeHandle((Handle) hPrint);

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

In addition to converting the print record, you must also translate QuickDraw data using the QuickDraw GX Translator. You call the `GXInstallQDTranslator` function to install the translator, and you call the `GXRemoveQDTranslator` function when you are finished with the translation. The QuickDraw GX Translator and these functions are described in the QuickDraw GX environment chapter of *Inside Macintosh: QuickDraw GX Environment and Utilities*.

After you call `GXInstallQDTranslator`, you proceed to print using the normal print loop; for example, by calling `GXStartPage` to start a new page and `GXFinishPage` to finish it. The results of any QuickDraw function is translated and the output is spooled. When you are finished printing, call `GXRemoveQDTranslator` to end translation.

Core Printing Features Reference

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

The “Constants and Data Types” section shows the Gestalt selector enumeration for QuickDraw GX printing features, the data types for QuickDraw GX printing-related objects, the Edit menu structure, and the dialog box result enumeration.

The “Functions” section describes functions for initializing and terminating printing features, handling errors, creating and managing job objects, printing using QuickDraw GX, obtaining information on printing-related objects, displaying the Page Setup and Print dialog boxes, and converting a print record into a job object.

The “Application-Defined Functions” section shows sample functions for flattening and unflattening job objects.

Constants and Data Types

This section describes the data types and constants that you use to initialize QuickDraw GX printing features, reference QuickDraw GX printing-related objects, and support QuickDraw GX print dialog boxes.

You can use the Gestalt selector enumeration to test for the existence of QuickDraw GX printing features.

The QuickDraw GX printing-related object structures are private. You can access print objects through references.

You can use the Edit menu structure to specify the location of the Edit menu and its menu items when displaying print dialog boxes.

You can use the dialog box result enumeration to store the user’s response to QuickDraw GX print dialog boxes.

Gestalt Selectors for Printing

To test for the existence of QuickDraw GX printing features, use the `Gestalt` function. The Gestalt selectors for the QuickDraw GX printing manager version and QuickDraw GX are defined as follows:

```
#define gestaltGXPrintingMgrVersion 'pmgr'
#define gestaltGXVersion           'qdgx'
```

The `Gestalt` function is discussed in *Inside Macintosh: Operating System Utilities*.

QuickDraw GX Printing-Related Objects

QuickDraw GX provides you with access to printing-related objects through references. The contents of the structures are private.

You access a job object through a job object reference:

```
typedef struct gxPrivateJobRecord *gxJob;
```

You access printer objects through a printer object reference:

```
typedef struct gxPrivatePrinterRecord *gxPrinter;
```

You access a format object through a format object reference:

```
typedef struct gxPrivateFormatRecord *gxFormat;
```

You access a paper-type object through a paper-type object reference:

```
typedef struct gxPrivatePaperTypeRecord *gxPaperType;
```

You access a print file object through a print file object reference:

```
typedef struct gxPrivatePrintFileRecord *gxPrintFile;
```

QuickDraw GX also provides the job, format, and paper-type collection objects. You access collection objects through a collection object reference:

```
typedef struct PrivateCollectionRecord *Collection;
```

Edit Menu Location

When displaying QuickDraw GX print dialog boxes, your application needs to specify the location of the Edit menu and its menu items. Your application specifies the location of the Edit menu and its menu items in the Edit menu structure. The Edit menu structure is defined as follows:

```
struct gxEditMenuRecord {
    short    editMenuID;
    short    cutItem;
    short    copyItem;
    short    pasteItem;
    short    clearItem;
    short    undoItem;
} gxEditMenuRecord;
```

Field descriptions

<code>editMenuID</code>	Your application's resource ID for the Edit menu.
<code>cutItem</code>	The position of the cut menu item under the Edit menu.
<code>copyItem</code>	The position of the copy menu item under the Edit menu.
<code>pasteItem</code>	The position of the paste menu item under the Edit menu.
<code>clearItem</code>	The position of the clear menu item under the Edit menu.
<code>undoItem</code>	The position of the undo menu item under the Edit menu.

Dialog Box Results

QuickDraw GX print dialog boxes support dialog box results. Results are defined in the dialog box result enumeration.

```
enum {
    gxCancelSelected  = (gxDialogResult) 0,
    gxOKSelected      = (gxDialogResult) 1,
    gxRevertSelected  = (gxDialogResult) 2
};

typedef long gxDialogResult;
```

Constant descriptions`gxCancelSelected`

Represents a cancelation of the dialog box without action being taken, such as when the user chooses Cancel or presses Escape while in a dialog box.

`gxOKSelected`

Represents a confirmation, such as when the user chooses Format in the Page Setup dialog box.

`gxRevertSelected`

Represents a request to undo one or more actions, such as when the user chooses Remove to remove a page format while in the Custom Page Setup dialog box.

Functions

This section describes the functions for initializing and terminating printing features, handling errors, creating and managing job objects, printing using QuickDraw GX, obtaining information on print objects, displaying print dialog boxes, and converting a print record into a job object.

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

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

IMPORTANT

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

Initializing and Terminating QuickDraw GX Printing Features

After you call the `GXEnterGraphics` function to initialize QuickDraw GX, you can call the `GXInitPrinting` function to initialize printing features within QuickDraw GX.

When you have successfully called the `GXInitPrinting` function and you need to terminate printing features within QuickDraw GX, you must call the `GXExitPrinting` function.

GXInitPrinting

You can use the `GXInitPrinting` function to initialize printing features within QuickDraw GX.

```
OSErr GXInitPrinting (void);
```

function result An error code of type `OSErr`.

DESCRIPTION

Before you call the `GXInitPrinting` function, you must call the `GXEnterGraphics` function to initialize QuickDraw GX. You should also use the `Gestalt` function to determine whether QuickDraw GX printing features are available on the user's system. The `Gestalt` selector is `'pmgr'`.

SPECIAL CONSIDERATIONS

If the `GXInitPrinting` function returns an error, you should not attempt to call other QuickDraw GX printing-related functions.

RESULT CODES

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

SEE ALSO

The `GXEnterGraphics` function that initializes QuickDraw GX is described in *Inside Macintosh: QuickDraw GX Environment and Utilities*.

To terminate printing features in QuickDraw GX, use the `GXExitPrinting` function, which is described in the next section.

GXExitPrinting

You can use the `GXExitPrinting` function to terminate printing features within QuickDraw GX.

```
OSErr GXExitPrinting (void);
```

function result An error code of type `OSErr`.

DESCRIPTION

The `GXExitPrinting` function terminates printing features within QuickDraw GX only after you have successfully called the `GXInitPrinting` function. You cannot call QuickDraw GX printing functions after you call the `GXExitPrinting` function.

You must call the `GXExitPrinting` function before you call the `GXExitGraphics` function to shut down QuickDraw GX.

Before you call the `GXExitPrinting` function, you should dispose of all QuickDraw GX printing-related objects. If you want to use these objects again, you should save them.

RESULT CODES

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

SEE ALSO

For information about saving printing-related objects, see “Saving a Job Object With a Document File” beginning on page 2-24.

The `GXExitGraphics` function is described in the environment chapter of *Inside Macintosh: QuickDraw GX Environment and Utilities*.

Handling Errors

QuickDraw GX printing features allow you to poll for errors in two ways: immediately after you call a function or after you call groups of functions. QuickDraw GX provides the `GXGetJobError` function to allow you to poll for errors in both ways.

To allow your application to manage separate documents, errors are local to a job object. To store an error with a particular job object, you use the `GXSetJobError` function.

GXGetJobError

You can use the `GXGetJobError` function to obtain the first error encountered for a particular job object since the last call to `GXGetJobError`.

```
OSErr GXGetJobError (gxJob aJob);
```

`aJob` A reference to the job object whose most recent error you want to obtain.

function result An error code of type `OSErr`.

DESCRIPTION

The `GXGetJobError` function returns printing-related errors associated with a job object. Initially, you can call this function to obtain the current error code. If you immediately call this function a second time, it returns `noErr`.

You can use the `GXSetJobError` function to store an error in a specific job object.

SPECIAL CONSIDERATIONS

After an error occurs, calls to QuickDraw GX printing-related functions associated with the specified job object return immediately without executing, until the `GXGetJobError` function is called.

The `GXGetJobError` function does *not* move Macintosh memory; however, your application should not call this function at interrupt time, because it relies on data structures that may move.

SEE ALSO

Error-handling methods using the `GXGetJobError` function are described in “Error Handling,” which begins on page 2-14.

The `GXSetJobError` function is described in the next section.

GXSetJobError

You can use the `GXSetJobError` function to store an error in the provided job object.

```
void GXSetJobError (gxJob aJob, OSErr anError);
```

`aJob` A reference to the job object in which to store the error.

`anError` The error to store.

DESCRIPTION

The `GXSetJobError` function stores an error with a particular job object. This function is useful when you want to abort or cancel spooling.

Most applications do not need to use this function because QuickDraw GX sets the error for you. You might want to use it, however, to artificially raise an error condition.

SPECIAL CONSIDERATIONS

An existing error is replaced when you call the `GXSetJobError` function. If you wish to save a previous error, you must call the `GXGetJobError` function to obtain an error prior to calling the `GXSetJobError` function.

RESULT CODES

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

Creating and Managing Job Objects

When a user creates a new document, you need to create a corresponding job object using the `GXNewJob` function. When a user closes a printable document, you need to dispose of its corresponding job object using the `GXDisposeJob` function.

When a user saves a printable document, you need to flatten its job object using either the `GXFlattenJobToHdl` function or the `GXFlattenJob` function. When a user opens a printable document, you need to retrieve its job object using either the `GXUnflattenJobFromHdl` function or the `GXUnflattenJob` function.

When you receive a resume event, you should use the `GXUpdateJob` function to update the job object because the printing environment may have changed.

GXNewJob

You can use the `GXNewJob` function to create a job object to associate with a printable document.

```
OSErr GXNewJob (gxJob *aJob);
```

`aJob` On return, a reference to the newly created job object.

function result An error code of type `OSErr`.

DESCRIPTION

The `GXNewJob` function allocates space for a job object and returns a reference to the job object. You need to call this function each time a user creates a new printable document.

When QuickDraw GX creates a new job object, it contains default values. Specifically, it contains a default format and a default paper type. The default format and default paper type are defined by the default output printer's printer driver. If there is no default output printer's printer driver, the job object uses the format and paper type associated with "Any Printer."

You should call the `GXInstallApplicationOverride` function after you call the `GXNewJob` function to support QuickDraw GX print dialog boxes.

When a user closes a document, you need to dispose of a job object using the `GXDisposeJob` function.

RESULT CODES

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

SEE ALSO

Listing 2-1 on page 2-12 shows how to use the `GXNewJob` function to create a job object for a printable document.

The `GXInstallApplicationOverride` function for supporting QuickDraw GX print dialog boxes is described on page 2-71.

To dispose of a job object, see the description of the `GXDisposeJob` function in the next section.

GXDisposeJob

You can use the `GXDisposeJob` function to dispose of a job object associated with a printable document.

```
OSErr GXDisposeJob (gxJob aJob);
```

`aJob` A reference to the job object to be disposed of.

function result An error code of type `OSErr`.

DESCRIPTION

You should call the `GXDisposeJob` function when a user closes a printable document and deallocates space for an existing job object. This function returns an error if the specified job object is `nil`.

Before you dispose of a job object, you should call the `GXFlattenJobToHdl` function or the `GXFlattenJob` function to save a job object when a user saves a document.

Core Printing Features

RESULT CODES

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

SEE ALSO

Listing 2-9 on page 2-29 shows how to use the `GXDisposeJob` function to dispose of a job object when a user closes a document.

The `GXFlattenJobToHdl` function for saving job objects in a handle is described in the next section. The `GXFlattenJob` function for saving job objects by calling a function is described on page 2-57.

GXFlattenJobToHdl

You can use the `GXFlattenJobToHdl` function to flatten a job object into a handle.

```
Handle GXFlattenJobToHdl (gxJob aJob, Handle aHandle);
```

`aJob` A reference to the job object to be flattened.

`aHandle` The handle into which the flattened data is placed.

function result The handle into which the flattened data is placed.

DESCRIPTION

The `GXFlattenJobToHdl` function provides your application with a mechanism for saving all information associated with a job object in a handle. You should call this function when a user saves a printable document.

You specify a handle in the `aHandle` parameter. QuickDraw GX grows or shrinks the size of the handle you provide to accommodate the size of the job object. You can specify `nil` in this parameter to allow QuickDraw GX to create and return a handle for you.

When you save a printable document, you can write the handle to the file's resource or data fork. You cannot directly modify the contents of this handle.

When a user opens a printable document, you need to unflatten all information associated with a job object using the `UnflattenJobToHdl` function.

If you do not wish to save data in a handle, you can also use the `GXFlattenJob` and `GXUnflattenJob` functions to specify a function to save and restore a job object.

RESULT CODES

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

SEE ALSO

Listing 2-7 on page 2-25 shows how to use the `GXFlattenJobToHdl` function to save a job object.

To unflatten all information associated with a job object, see the `GXUnflattenJobFromHdl` function, which is described on page 2-58.

You can also specify a function to save information associated with a job object by using the `GXFlattenJob` function, which is described in the next section.

GXFlattenJob

You can use the `GXFlattenJob` function when you want to call a function to flatten a job object.

```
void GXFlattenJob (gxJob aJob,
                  gxPrintingFlattenProc aPrintingFlattenProc,
                  void *aVoid);
```

`aJob` A reference to the job object to be flattened.

`aPrintingFlattenProc`
 A pointer to a flattening function.

`aVoid` A reference variable passed to the flattening function.

DESCRIPTION

The `GXFlattenJob` function provides your application with a mechanism for saving all information associated with a job object by specifying a pointer to a flattening function. QuickDraw GX calls your flattening function multiple times as it saves job object-related data to disk.

You specify a pointer to a flattening function in the `aPrintingFlattenProc` parameter of the `GXFlattenJob` function. You may prefer to use the `GXFlattenJob` function (instead of the `GXFlattenJobToHdl` function) because it requires less memory to save portions of job object data to disk than it does to save all the data in a single handle.

When a user opens a printable document, you need to unflatten all information associated with a job object using the `GXUnflattenJob` function.

Core Printing Features

RESULT CODES

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

SEE ALSO

Listing 2-8 on page 2-28 shows how to use the `GXFlattenJob` function to save a job object.

An example of a flattening function is described on page 2-76.

To unflatten all information flattened using `GXFlattenJob`, see the `GXUnflattenJob` function, which is described on page 2-59.

To flatten a job to a handle, see the `GXFlattenJobToHdl` function, which is described on page 2-56.

GXUnflattenJobFromHdl

You can use the `GXUnflattenJobFromHdl` function to unflatten a job object that you previously flattened using the `GXFlattenJobToHdl` function.

```
gxJob GXUnflattenJobFromHdl (gxJob aJob, Handle aHandle);
```

`aJob` A reference to the job object into which unflattened data is placed.

`aHandle` A handle from which the job object is to be read.

function result The unflattened job object.

DESCRIPTION

The `GXUnflattenJobFromHdl` function provides your application with a mechanism for retrieving all information associated with a job object from a handle. You should call this function when a user opens a printable document containing a job object that was previously flattened using the `GXFlattenJobToHdl` function.

In the `aJob` parameter, you specify a job object in which to place the unflattened job object data. You can specify `nil` in this parameter to allow QuickDraw GX to create and return a job object for you.

The `aHandle` parameter specifies the handle from which the job object information is read. You previously specified this handle using the `GXFlattenJobToHdl` function.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
<code>gxFlattenVersionTooNew</code>	An attempt to unflatten a job object that was flattened using a later version of QuickDraw GX.
<code>gxPaperTypeNotFound</code>	The paper-type object cannot be located.
<code>collectionVersionErr</code>	Version of the collection object is not compatible with the current version of the Collection Manager.

SEE ALSO

Listing 2-10 on page 2-30 shows how to use the `GXUnflattenJobFromHdl` function to retrieve a job object from a handle.

You specify a handle in which to save a job object using the `GXFlattenJobToHdl` function, which is described on page 2-56.

GXUnflattenJob

You can use the `GXUnflattenJob` function to unflatten a job object that you previously flattened using the `GXFlattenJob` function.

```
gxJob GXUnflattenJob (gxJob aJob,
                     gxPrintingFlattenProc aPrintingFlattenProc,
                     void *aVoid);
```

`aJob` A reference to the job object to be unflattened.

`aPrintingFlattenProc`
 A pointer to a flattening function.

`aVoid` A reference variable passed to the flattening function.

function result The unflattened job object.

DESCRIPTION

The `GXUnflattenJob` function provides your application with a mechanism for retrieving all information associated with a job object by executing an application-supplied function. In the `aPrintingFlattenProc` parameter, you specify a pointer to an unflattening function. QuickDraw GX calls your unflattening function multiple times as it retrieves job object-related data from disk.

In the `aJob` parameter, you specify a job object in which to place the unflattened job object data. You can specify `nil` in this parameter to allow QuickDraw GX to create and return a job object for you.

Core Printing Features

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
<code>gxFlattenVersionTooNew</code>	An attempt to unflatten a job object that was flattened using a later version of QuickDraw GX.
<code>gxPaperTypeNotFound</code>	The paper-type object cannot be located.
<code>collectionVersionErr</code>	Version of the collection object is not compatible with the current version of the Collection Manager.

SEE ALSO

Listing 2-10 on page 2-30 shows an example of how to use the `GXUnflattenJob` function.

You specify a function to save a job object by using the `GXFlattenJob` function, which is described on page 2-57.

GXUpdateJob

You can use the `GXUpdateJob` function to update the contents of a job object.

```
Boolean GXUpdateJob (gxJob aJob);
```

`aJob` A reference to the job object whose contents may need to change.

function result A Boolean, which returns `true` if anything actually changed.

DESCRIPTION

The `GXUpdateJob` function updates the job object to reflect the current QuickDraw GX environment. You must call this function when your application receives a resume event, indicating that it had been switched out because the user may have changed the characteristics of a printer. For example, the user may have added an extension while the application was switched out.

RESULT CODES

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

SEE ALSO

For an example that uses the `GXUpdateJob` function, see “Updating Job Object Information” on page 2-42.

Printing With QuickDraw GX

To support printing from the Finder, your application needs to call the `GXSelectJobOutputPrinter` function to specify an output printer.

When the user requests printing, you should call the `GXGetJobPageRange` function to obtain the user-specified page range.

You call the `GXStartJob` function to begin printing a document. If your application stores each page as a single picture shape, you should use the `GXPrintPage` function to print each page in a document.

You may also choose to use the `GXStartPage`, `GXDrawShape`, and `GXFinishPage` functions to draw and print data. You should use these functions if your application does not store each page as a single picture shape.

After you have finished calling the `GXPrintPage` or the `GXFinishPage` function (depending on the approach you choose), you call the `GXFinishJob` function to tell QuickDraw GX that the document is ready to be spooled for printing in the background.

The `GXDrawShape` function is described in the shape objects chapter of *Inside Macintosh: QuickDraw GX Objects*.

GXSelectJobOutputPrinter

You can use the `GXSelectJobOutputPrinter` function to specify an output printer for a printable document.

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

<code>aJob</code>	A reference to the job object for which you are specifying an output printer.
-------------------	---

<code>printerName</code>	The name of the desktop printer.
--------------------------	----------------------------------

Core Printing Features

DESCRIPTION

Your application is responsible for determining the output printer on which to print the document.

For example, when the user selects and prints a document from the Finder, your application needs to respond to the Print Documents ('pdoc') Apple event and then call the `GXSelectJobOutputPrinter` function to specify an output printer on which to print the selected document. The printer name can be obtained by using the Apple event's optional attribute, `keyOptionalKeywordAttr`.

RESULT CODES

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

SEE ALSO

Listing 2-16 on page 2-40 shows how to respond to the Print Documents Apple event and use the `GXSelectJobOutputPrinter` function to specify an output printer.

GXGetJobPageRange

You can use the `GXGetJobPageRange` function to obtain a user-specified page range.

```
void GXGetJobPageRange (gxJob aJob, long *firstPage,
                        long *lastPage);
```

<code>aJob</code>	A reference to the job object for which to retrieve the page range.
<code>firstPage</code>	On return, the first page the user wants to print.
<code>lastPage</code>	On return, the last page the user wants to print.

DESCRIPTION

When the user requests printing, you should call the `GXGetJobPageRange` function to obtain the user-specified page range. The user specifies a page range in the Print dialog box.

You can set the `firstPage` parameter or the `lastPage` parameter to `nil` to ignore the result.

RESULT CODES

<code>gxSegmentLoadFailedErr</code>	A required code segment could not be found, or there was not enough memory to load it.
<code>collectionItemNotFoundErr</code>	The collection object item cannot be located.

SEE ALSO

Listing 2-5 on page 2-21 and Listing 2-6 on page 2-23 show how to use the `GXGetJobPageRange` function to obtain the user-specified page range.

GXStartJob

You can use the `GXStartJob` function to initiate printing when a user wants to print a document.

```
void GXStartJob (gxJob aJob, StringPtr docName, long pageCount);
```

<code>aJob</code>	A reference to the job object of the print job to print.
<code>docName</code>	The name of the document to print.
<code>pageCount</code>	The number of pages to print.

DESCRIPTION

You use the `GXStartJob` function to begin printing a document. In the `aJob` parameter, you specify the job object associated with the document to print. In the `docName` parameter, you specify the name of the user's document. You can set this parameter to `nil` to use the default document name.

In the `pageCount` parameter, you specify the total number of pages the user chose to print or pass 0 if the page count is unknown. You can call the `GXGetJobPageRange` function to obtain the page range. In response to the `GXStartJob` call, QuickDraw GX displays the current page and the print job's page count, if it is known, in the Status dialog box.

SPECIAL CONSIDERATIONS

Immediately after you call the `GXStartJob` function, you should check for errors by calling the `GXGetJobError` function. Only if no errors are returned should you call the `GXFinishJob` function.

Core Printing Features

RESULT CODES

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

SEE ALSO

Listing 2-5 on page 2-21 and Listing 2-6 on page 2-23 show how to use the `GXStartJob` function to begin printing a document.

For information about the `GXGetJobPageRange` function, see the previous section.

The `GXGetJobError` function is described on page 2-52. The `GXFinishJob` function is described on page 2-65.

GXPrintPage

You can use the `GXPrintPage` function to print a page in a document if your application stores each page as a single picture shape.

```
void GXPrintPage (gxJob aJob, long pageNumber, gxFormat aFormat,
                  gxShape aPage);
```

<code>aJob</code>	A reference to the job object whose page you want to print.
<code>pageNumber</code>	The page number for the page.
<code>aFormat</code>	A reference to the format object for the page.
<code>aPage</code>	A reference to the picture shape that specifies the output for the page.

DESCRIPTION

The `GXPrintPage` function prints a page of a document. In the `aPage` parameter, you specify the picture shape for each page. In the `pageNumber` parameter, you set the page to print. QuickDraw GX compares the specified page number with the page range chosen by the user and spools the page if it is within the page range. If it is not within the range, QuickDraw GX ignores the data.

In the `aFormat` parameter, you specify the format object for the page. You need to provide your own mechanism for associating individual document pages with format objects.

You should loop through each page of a document, calling the `GXPrintPage` function for each page's picture shape. You should check for errors after you print each page and exit the loop if necessary.

If your application does not store each page as a single picture shape, you should use the `GXStartPage`, `GXDrawShape`, and `GXFinishPage` functions to print the page.

RESULT CODES

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

SEE ALSO

Listing 2-5 on page 2-21 shows how to use the `GXPrintPage` function to print each page of a document.

Picture shapes are discussed in *Inside Macintosh: QuickDraw GX Graphics*.

The `GXStartPage` function is described on page 2-66. The `GXFinishPage` function is described on page 2-67. The `GXDrawShape` function is described in the shape objects chapter of *Inside Macintosh: QuickDraw GX Objects*.

In addition to the result codes listed above, you may also receive errors that can occur while flattening graphics objects during spooling. For more information about the spooling phase of printing, see the chapter “Introduction to Printing With QuickDraw GX” in this book. Flattening graphics objects is described in the shape objects chapter of *Inside Macintosh: QuickDraw GX Objects*.

GXFinishJob

You can use the `GXFinishJob` function to notify QuickDraw GX that printing is complete.

```
void GXFinishJob (gxJob aJob);
```

`aJob` A reference to the job object being printed.

DESCRIPTION

The `GXFinishJob` function completes the application phase of printing. You should call this function after you have called the `GXPrintPage` function to print each page in a document.

SPECIAL CONSIDERATIONS

You should only call the `GXFinishJob` function if the `GXStartJob` function doesn't return errors.

Core Printing Features

RESULT CODES

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

In addition to the result codes listed above, you may also receive errors that can occur while flattening graphics objects during spooling. Flattening graphics objects is described in the shape objects chapter of *Inside Macintosh: QuickDraw GX Objects*.

SEE ALSO

Listing 2-5 on page 2-21 and Listing 2-6 on page 2-23 show how to use the `GXFinishJob` function to tell QuickDraw GX that the document is ready to be queued for printing in the background.

The `GXStartJob` function is described on page 2-63.

Phases of printing are described in the chapter “Introduction to Printing With QuickDraw GX” in this book.

GXStartPage

You can use the `GXStartPage` function to print each page in a document if your application does not store each page as a single picture shape.

```
Boolean GXStartPage (gxJob aJob, long pageNumber,
                    gxFormat aFormat, long numViewPorts,
                    gxViewport *viewPortList);
```

<code>aJob</code>	A reference to the job object being printed.
<code>pageNumber</code>	The page number of the page being printed.
<code>aFormat</code>	A reference to the format object for the page.
<code>numViewPorts</code>	The number of view ports contained in the <code>viewPortList</code> parameter.
<code>viewPortList</code>	A pointer to the list of references to view ports to use to capture shapes.

function result Returns `true` if the page you specify in the `pageNumber` parameter is within the user-specified page range, `false` if the page you specify is not.

DESCRIPTION

You use the `GXStartPage` function to start printing the shapes drawn with `GXDrawShape`. You call the `GXStartPage` function after you call the `GXStartJob` function.

Core Printing Features

In the `GXStartPage` function, you specify in the `pageNumber` parameter the page number of the page to print. QuickDraw GX compares the specified page number with the page range. The `GXStartPage` function returns `true` if the page you specify is within the user-specified page range, and returns `false` if it is not. You can call the `GXGetJobPageRange` function to determine the range of pages.

In the `viewPortList` parameter, you specify the view ports to use to capture shapes. The part of the shape that is drawn through a view port is printed. In the `numViewPorts` parameter, you specify the number of view ports in the `viewPortList` parameter.

SPECIAL CONSIDERATIONS

After you finish calling the `GXStartPage` function, you should immediately check for errors using the `GXGetJobError` function. Only if no errors are returned should you draw the page's shapes and call the `GXFinishPage` function.

RESULT CODES

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

SEE ALSO

Listing 2-6 on page 2-23 shows how to use the `GXStartPage` function to print each page of a document.

The `GXDrawShape` function is discussed in the shape objects chapter of *Inside Macintosh: QuickDraw GX Objects*.

View port objects are discussed in the view-related objects chapter of *Inside Macintosh: QuickDraw GX Objects*.

The `GXFinishPage` function is described in the next section. The `GXGetJobError` function is described on page 2-52. The `GXGetJobPageRange` function is described on page 2-62.

GXFinishPage

You can use the `GXFinishPage` function to notify QuickDraw GX that you have finished capturing shapes for the page.

```
void GXFinishPage (gxJob aJob);
```

`aJob` A reference to the job object being printed.

Core Printing Features

DESCRIPTION

You should call the `GXFinishPage` function after you have finished drawing the data for a page using the `GXDrawShape` function. In the `aJob` parameter, you specify the job object being printed.

After you call the `GXFinishPage` function for the final page to be printed, call the `GXFinishJob` function to notify QuickDraw GX that printing is complete.

SPECIAL CONSIDERATIONS

You should only call the `GXFinishPage` function if the `GXStartPage` function doesn't return errors.

RESULT CODES

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

In addition to the following result codes, you may also receive errors that can occur while flattening graphics objects during spooling. Flattening graphics objects is described in the shape objects chapter of *Inside Macintosh: QuickDraw GX Objects*.

SEE ALSO

Listing 2-6 on page 2-23 shows how to use the `GXFinishPage` function to tell QuickDraw GX that you have finished capturing shapes.

The `GXStartPage` function is described in the previous section.

The `GXDrawShape` function is discussed in the shape objects chapter of *Inside Macintosh: QuickDraw GX Objects*.

Obtaining Information on Printing-Related Objects

QuickDraw GX functions allow you to obtain basic information about the job object and format objects associated with a printable document. Although a document can contain multiple format objects, all documents contain at least one format object, called the default format.

When a user wants to print a document, you should call the `GXGetJobFormat` function to access the format objects associated with a particular job object.

You can specify a format object in the `GXGetFormatJob` function to obtain the job object that references this format object.

You use the `GXGetFormatDimensions` function to obtain the dimensions information from a format object. The information includes the physical dimensions of the paper (the paper size) and the printable area within these dimensions (the page size) after scaling and orientation have been applied.

GXGetJobFormat

You can use the `GXGetJobFormat` function to obtain the format objects associated with a job object.

```
gxFormat GXGetJobFormat (gxJob aJob, long whichFormat);
```

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

`whichFormat` The index of the format object to retrieve.

function result A reference to a format object.

DESCRIPTION

The `GXGetJobFormat` function allows you to obtain a format object from the job object specified in the `aJob` parameter. The `whichFormat` parameter specifies the format object to return. You can set this parameter to 1 to obtain the default format. The default format is defined by the formatting printer.

RESULT CODES

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

SEE ALSO

Listing 2-5 on page 2-21 and Listing 2-6 on page 2-23 show how to use the `GXGetJobFormat` function to obtain the default format when a user wants to print a document.

Manipulating format objects is described in the chapter “Page Formatting and Dialog Box Customization” in this book.

GXGetFormatJob

You can use the `GXGetFormatJob` function to obtain the job object associated with a format object.

```
gxJob GXGetFormatJob (gxFormat aFormat);
```

`aFormat` A reference to the format object whose job object you wish to obtain.

function result A reference to a job object.

Core Printing Features

DESCRIPTION

In the `GXGetFormatJob` function, you specify a format object whose job object you want to obtain. You specify the format object in the `aFormat` parameter. You should call this function when you have a format object reference and you want to obtain the reference of the job object associated with it.

RESULT CODES

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

GXGetFormatDimensions

You can use the `GXGetFormatDimensions` function to obtain the page size and paper size associated with a format object.

```
void GXGetFormatDimensions (gxFormat aFormat,
                           gxRectangle *pageSize,
                           gxRectangle *paperSize);
```

<code>aFormat</code>	A reference to the format object whose dimensions you wish to obtain.
<code>pageSize</code>	On return, the imageable area—the area inside the margins where shapes may be drawn.
<code>paperSize</code>	On return, the physical dimensions of the paper.

DESCRIPTION

The `GXGetFormatDimensions` function returns a page size and paper size associated with a format object, after scaling and orientation have been applied. This function provides your application with boundary information that is useful for setting up margins for the drawing areas in your application. It is also useful for setting up rulers in your application to display to users.

You can specify `nil` in either the `pageSize` or `paperSize` parameters if you are interested in only one of the values.

The page size is anchored at location (0.0, 0.0), regardless of orientation or scaling. The paper size is outset from the page size, and the coordinates for the top-left corner of the paper are negative. Because the page coordinates are zero-based, you can start drawing at (0.0, 0.0) without regard for the paper size.

RESULT CODES

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

SEE ALSO

For a description of format object mapping and how it affects the dimensions property, see the chapter “Page Formatting and Dialog Box Customization” in this book.

Displaying the Page Setup and Print Dialog Boxes

To support QuickDraw GX print dialog boxes, your application must override the `gxPrintingEvent` message by installing an override function with the `GXInstallApplicationOverride` function.

When the user chooses the Page Setup menu command from the File menu, you call the `GXJobDefaultFormatDialog` function to display the Page Setup dialog box.

When the user chooses the Print menu command from the File menu, you call the `GXJobPrintDialog` function to display the Print dialog box.

GXInstallApplicationOverride

You can use the `GXInstallApplicationOverride` function to override messages QuickDraw GX sends to your application.

```
void GXInstallApplicationOverride (gxJob aJob, short messageID,
                                  void *override);
```

`aJob` A reference to the job object into which to install the override.

`messageID` The ID of the message to override.

`override` A pointer to a function with which to override a message.

DESCRIPTION

You can use the `GXInstallApplicationOverride` function to specify a function that is called in response to the message specified in the `messageID` parameter. For example, you can override the `gxPrintingEvent` message that QuickDraw GX sends to your application each time it receives an event by specifying a function to call in the `override` parameter.

You specify a pointer to an override function in the `override` parameter. Set this parameter to `nil` to remove your application’s override of a message.

Core Printing Features

RESULT CODES

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

SEE ALSO

Listing 2-1 on page 2-12 shows how to override the `gxPrintingEvent` message using the `GXInstallApplicationOverride` function.

Supporting QuickDraw GX dialog boxes is discussed in “Supporting QuickDraw GX Print Dialog Boxes,” which begins on page 2-17.

GXJobDefaultFormatDialog

You can use the `GXJobDefaultFormatDialog` function to display the Page Setup dialog box.

```
gxDialogResult GXJobDefaultFormatDialog (gxJob aJob,
                                         gxEditMenuRecord *anEditMenuRecord);
```

<code>aJob</code>	A reference to the job object whose default format you are allowing the user to modify.
-------------------	---

<code>anEditMenuRecord</code>	A pointer to the Edit menu structure.
-------------------------------	---------------------------------------

function result The user’s response to the dialog box.

DESCRIPTION

After you use the `GXJobDefaultFormatDialog` function to display the Page Setup dialog box, the user can specify formatting information for the default format. For example, the user can specify the paper size, orientation, and the default formatting printer.

In the `anEditMenuRecord` parameter you specify an Edit menu structure to support the standard editing operations of cut, copy, paste, and clear in dialog boxes.

The function returns `gxOKSelected` if Format is chosen or `gxCancelSelected` if Cancel is chosen.

If an error occurs, the function returns `gxCancelSelected`. Call the `GXGetJobError` function to determine which error occurred.

This function causes QuickDraw GX to send the `gxJobDefaultFormatDialog` message, which you can override to customize the Page Setup dialog box.

Note that QuickDraw GX stores a user’s responses to some dialog items in the Page Setup dialog box in a format collection.

RESULT CODES

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

SEE ALSO

Listing 2-14 on page 2-36 shows how to use the `GXJobDefaultFormatDialog` function to display the Page Setup dialog box.

The Edit menu structure is described on page 2-48.

The dialog box result enumeration is described on page 2-48.

The format collection is discussed in the chapter “Page Formatting and Dialog Box Customization” in this book.

GXJobPrintDialog

You can use the `GXJobPrintDialog` function to display the Print dialog box when the user chooses the Print menu command from the File menu.

```
gxDialogResult GXJobPrintDialog (gxJob aJob,
                                gxEditMenuRecord *anEditMenuRecord);
```

<code>aJob</code>	A reference to the job object whose print settings you are allowing the user to modify.
-------------------	---

<code>anEditMenuRecord</code>	A pointer to the Edit menu structure.
-------------------------------	---------------------------------------

function result The user’s response to the dialog box.

DESCRIPTION

After you use the `GXJobPrintDialog` function to display the Print dialog box, the user can specify information related to actual printing of the document. For example, the user can specify the printer, print quality, number of copies to print, page range, automatic or manual paper feed, and whether a document should be output to a printer or a file.

A user must select an output printer in the Print dialog box regardless of the formatting printer specified in the Page Setup dialog box. The output printer does not need to be in the same device class as the printer for which the document is formatted.

Core Printing Features

In the `anEditMenuRecord` parameter you specify an Edit menu structure. Your application specifies the location of the Edit menu and its menu items in the Edit menu structure.

The function returns `gxOKSelected` if Print is chosen or `gxCancelSelected` if Cancel is chosen.

If an error occurs, the function returns `gxCancelSelected`. Call the `GXGetJobError` function to determine which error occurred.

This function causes QuickDraw GX to send the `gxJobPrintDialog` message, which you can override to customize the Print dialog box.

QuickDraw GX stores a user's responses to some items in the Print dialog box in the job collection.

RESULT CODES

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

SEE ALSO

Listing 2-15 on page 2-39 shows how to use the `GXJobPrintDialog` function to display the Print dialog box.

The Edit menu structure is described on page 2-48.

The dialog box result enumeration is described on page 2-48.

The job collection is discussed in the chapter "Page Formatting and Dialog Box Customization" in this book.

Converting a Print Record

QuickDraw GX allows documents originally created to print with the Macintosh Printing Manager to be printed by applications that support QuickDraw GX. Before a user can print these documents, you must convert the document's print record information into a job object using the `GXConvertPrintRecord` function.

GXConvertPrintRecord

You can use the `GXConvertPrintRecord` function to translate a print record into a job object.

```
void GXConvertPrintRecord (gxJob aJob, THPrint aPrint);
```

`aJob` A reference to the job object to receive the converted data.

`aPrint` The print record to be converted.

DESCRIPTION

QuickDraw GX copies contents of the specified print record into the specified job object. Before you call the `GXConvertPrintRecord` function, you must first allocate space for the job object using the `GXNewJob` function. QuickDraw GX attempts to preserve as much print record information as possible.

In addition to converting the print record, you must also translate QuickDraw data by calling the QuickDraw GX Translator functions, `GXInstallQDTranslator` and `GXRemoveQDTranslator`, or by calling the `GXConvertPICTToShape` function.

RESULT CODES

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

SEE ALSO

Listing 2-18 on page 2-45 shows how to use the `GXConvertPrintRecord` function to convert a print record into a job object.

The `GXNewJob` function is described on page 2-54.

The QuickDraw GX Translator functions, `GXInstallQDTranslator` and `GXRemoveQDTranslator`, are discussed in the environment chapter of *Inside Macintosh: QuickDraw GX Environment and Utilities*.

For information about the `GXConvertPICTToShape` function, see the environment chapter of *Inside Macintosh: QuickDraw GX Environment and Utilities*.

Application-Defined Functions

The following sections describe application-defined functions that implement message overrides and application-defined functions that flatten or unflatten job objects.

Message Override Functions

The `GXPrintingEvent` function specifies the declaration for a function that you must provide in order to respond to `gxPrintingEvent` messages.

GXPrintingEvent

You must install an override function that QuickDraw GX invokes in response to the `gxPrintingEvent` message. Your override 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 the `gxPrintingEvent` message to handle events, such as window update events, that occur during display of print dialog boxes. You cannot name your function `GXPrintingEvent`.

The default implementation of this message does nothing. You must override this message to correctly support print dialog boxes.

The `anEventRecord` parameter is a pointer to the event record. The event record contains information about what type of event occurred (a mouse-down, update, or key-down event, for example) and contains additional information associated with the event (for example, for a key-down event, the Event Manager also reports which key was pressed).

SPECIAL CONSIDERATIONS

You never send the `gxPrintingEvent` message yourself.

You typically create a total override of the `gxPrintingEvent` message.

RESULT CODES

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

SEE ALSO

Overriding the `gxPrintingEvent` message is described in “Supporting QuickDraw GX Print Dialog Boxes,” which begins on page 2-17.

The `GXInstallApplicationOverride` function is described on page 2-71.

The Event Manager, the `EventRecord` data type, and the `DialogSelect` function are discussed in the chapter “Event Manager” in *Inside Macintosh: Macintosh Toolbox Essentials*.

Flattening and Unflattening Functions for Job Objects

When a user saves or opens a printable document, you need to save or retrieve its corresponding job object. To save a job object, you can flatten it using the `GXFlattenJob` function. To retrieve a job object, you can unflatten it using the `GXUnflattenJob` function. In each of these functions you must provide a pointer to an application-supplied flattening or unflattening function, as appropriate. The following sections describe these flattening and unflattening functions.

MyFlattenFunction

To save a job object when a user saves a printable document, provide a pointer to an application-supplied flattening function in the `GXFlattenJob` function. The application-supplied function must match the following declaration. For example, this is how you should declare the function if you were to name it `MyFlattenFunction`:

```
OSErr MyFlattenFunction (long size, void *data, void *refCon);
```

<code>size</code>	The size of the segment (in bytes) to write.
<code>data</code>	A pointer to job object data to flatten.
<code>refCon</code>	A pointer to a reference constant for application-specific information.

function result An error code of type `OSErr`.

Core Printing Features

DESCRIPTION

When you use the `GXFlattenJob` function, QuickDraw GX calls your flattening function multiple times as it flattens job object data to disk. Each time it calls your function, the function should write the next segment of the job object until the entire job object is saved. You can use the `refCon` parameter to hold the file reference number of the file containing the data to flatten. You can return any `OSErr` value.

SEE ALSO

Listing 2-8 on page 2-28 shows how to use a flattening function.

The `GXFlattenJob` function is described on page 2-57.

To retrieve a job object that has been flattened, see the next section.

MyUnflattenFunction

To retrieve a job object when a document is opened, you can call the `GXUnflattenJob` function and provide a pointer to the application-supplied unflattening function you want to use. The application-supplied function must match the following declaration. For example, this is how you should declare the function if you were to name it `MyUnflattenFunction`:

```
OSErr MyUnflattenFunction (long size, void *data, void *refCon);
```

`size` The size of the segment (in bytes) to read.

`data` A pointer to job object data to unflatten.

`refCon` A pointer to a reference constant for application-specific information.

function result An error code of type `OSErr`.

DESCRIPTION

When you use the `GXUnflattenJob` function, QuickDraw GX calls your unflattening function multiple times as the unflattening function retrieves the job object data from disk. It continues to call your function until the entire job object is retrieved. You can use the `refCon` parameter to hold the file reference number of the file containing the data to unflatten. You can return any `OSErr` value.

SEE ALSO

Listing 2-11 on page 2-32 shows how to use an unflattening function.

The `GXUnflattenJob` function is described on page 2-59.

Summary of Core Printing Features

Constants and Data Types

Gestalt Selectors for Printing

```
#define gestaltGXPrintingMgrVersion 'pmgr'
#define gestaltGXVersion            'qdgx'
```

QuickDraw GX Printing-Related Objects

```
/* printing-related object structures */
typedef struct gxPrivateJobRecord *gxJob;           /* job object structure */
typedef struct gxPrivatePrinterRecord *gxPrinter;   /* printer object */
                                                    /* structure */
typedef struct gxPrivateFormatRecord *gxFormat;     /* format object */
                                                    /* structure */
typedef struct gxPrivatePaperTypeRecord *gxPaperType; /* paper-type object */
                                                    /* structure */
typedef struct gxPrivatePrintFileRecord *gxPrintFile; /* print file object */
                                                    /* structure */
typedef struct PrivateCollectionRecord *Collection; /* collection object */
                                                    /* structure */
```

Edit Menu Record Structure

```
typedef struct { /* location of Edit menu and its menu items */
    short editMenuID; /* resource ID of the Edit menu */
    short cutItem;    /* location of the cut menu item */
    short copyItem;   /* location of the copy menu item */
    short pasteItem;  /* location of the paste menu item */
    short clearItem;  /* location of the clear menu item */
    short undoItem;   /* location of the undo menu item */
} gxEditMenuRecord;
```

Dialog Box Results

```
typedef long gxDialogResult; /* dialog result data type */

/* dialog box result enumeration */
```

Core Printing Features

```
enum {
    gxCancelSelected = (gxDialogResult) 0, /* user canceled dialog box */
    gxOKSelected      = (gxDialogResult) 1, /* user confirmed dialog box */
    gxRevertSelected  = (gxDialogResult) 2 /* user chose Remove from
                                           the Custom Page Setup
                                           dialog box */
};
```

Functions

Initializing and Terminating QuickDraw GX Printing Features

```
OSErr GXInitPrinting      (void);
OSErr GXExitPrinting      (void);
```

Handling Errors

```
OSErr GXGetJobError       (gxJob aJob);
void GXSetJobError        (gxJob aJob, OSErr anError);
```

Creating and Managing Job Objects

```
OSErr GXNewJob            (gxJob *aJob);
OSErr GXDisposeJob        (gxJob aJob);
Handle GXFlattenJobToHdl   (gxJob aJob, Handle aHandle);
void GXFlattenJob          (gxJob aJob,
                           gxPrintingFlattenProc aPrintingFlattenProc,
                           void *aVoid);
gxJob GXUnflattenJobFromHdl (gxJob aJob, Handle aHandle);
gxJob GXUnflattenJob       (gxJob aJob,
                           gxPrintingFlattenProc aPrintingFlattenProc,
                           void *aVoid);
Boolean GXUpdateJob        (gxJob aJob);
```

Printing With QuickDraw GX

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

void GXGetJobPageRange        (gxJob aJob, long *firstPage, long *lastPage);
void GXStartJob               (gxJob aJob, StringPtr docName, long pageCount);
void GXPrintPage               (gxJob aJob, long pageNumber, gxFormat aFormat,
                               gxShape aPage);
```

Core Printing Features

```

void GXFinishJob          (gxJob aJob);
Boolean GXStartPage       (gxJob aJob, long pageNumber, gxFormat aFormat,
                           long numViewPorts, gxViewPort *viewPortList);
void GXFinishPage         (gxJob aJob);

```

Obtaining Information on Printing-Related Objects

```

gxFormat GXGetJobFormat   (gxJob aJob, long whichFormat);
gxJob GXGetFormatJob      (gxFormat aFormat);
void GXGetFormatDimensions (gxFormat aFormat, gxRectangle *pageSize,
                           gxRectangle *paperSize);

```

Displaying the Page Setup and Print Dialog Boxes

```

void GXInstallApplicationOverride
                           (gxJob, aJob, short messageID,
                           void *override);

gxDialogResult GXJobDefaultFormatDialog
                           (gxJob aJob,
                           gxEditMenuRecord *anEditMenuRecord);

gxDialogResult GXJobPrintDialog
                           (gxJob aJob,
                           gxEditMenuRecord *anEditMenuRecord);

```

Converting a Print Record

```

void GXConvertPrintRecord (gxJob aJob, THPrint aPrint);

```

Application-Defined Functions

Message Override Functions

```

OSErr GXPrintingEvent     (EventRecord *anEventRecord,
                           Boolean filterEvent);

```

Flattening and Unflattening Functions for Job Objects

```

OSErr MyFlattenFunction   (long size, void *data, void *refCon);
OSErr MyUnflattenFunction (long size, void *data, void *refCon);

```

