

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

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

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

You need to read this chapter if you are developing a printing extension or a printer driver for use with QuickDraw GX. This chapter constitutes the complete reference guide to the messages that you can use to develop these programs.

This chapter begins with a brief overview of how QuickDraw GX uses message passing to implement printing and then describes

- the terminology used to describe printing messages
- the categories of printing messages and which messages fit into each category
- the data structures that are used by the printing messages
- each of the messages that you can use to develop printing extensions and printer drivers

## About the Printing Messages

---

The printing messages described in this chapter are sent by QuickDraw GX during the process of printing a document. QuickDraw GX provides default implementations of many of these messages that perform the basic task for which the message is intended. This means that you only need to override those messages that apply to the specifics of your extension or driver. And many of the specifics of a device are defined in the resources that are included in your extension or driver.

For example, since QuickDraw GX provides robust default implementations, you can create a printer driver for a PostScript printer by defining resources and overriding only one message, the `GXSetupImageData` message. You need to override many messages for some printers and only a few messages for others; however, in most cases the default implementation supplied by QuickDraw GX provides functionality that you want to use. You can override messages to add to or modify that functionality as you need to for your task or device.

## Printing Message Terminology

---

This chapter uses object-oriented programming terminology to describe how the printing messages operate. This section reviews the terms used in this chapter to describe the messages. For a more complete description of these terms and how message-passing works in QuickDraw GX, read the chapter “Message Manager” in *Inside Macintosh: QuickDraw GX Environment and Utilities*. In this book, the chapter “Introduction to Printing Extensions and Drivers” contains an overview of how message-passing works with printing extensions and printer drivers.

The QuickDraw GX printing architecture uses a number of different object types to provide printing. Each print object is an abstract data type that encapsulates certain properties that you can use QuickDraw GX functions to access. The objects used in the QuickDraw GX printing architecture are described in *Inside Macintosh: QuickDraw GX Printing*. These objects include job objects, printer objects, format objects, and paper-type objects.

When QuickDraw GX needs to accomplish a printing task, it sends a message. For example, QuickDraw GX sends a message when it needs a driver to establish a communications connection with a printing device, when it wants to print a page, and when it needs a printer driver to convert a bitmap into a data format for a raster printer.

You respond to messages by overriding them. Your override of a message intercepts the message and takes some action. A number of message handlers can respond to each message, including the system software, the application program, a printer driver, and one or more printing extensions.

QuickDraw GX provides a default implementation of each printing messages. You can partially override a message to add to the default implementation’s response or to change the results of the default implementation. Depending on the nature of a message, you can forward it to the other handlers and then perform your tasks, or you can perform your tasks and then forward it to other handlers.

### Note

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

You can totally override some messages, which means that your override completely replaces the default implementation and does not forward the message. The “Special Considerations” section for each of the messages described in the section “Printing Messages Reference” in this chapter indicates which printing messages you can totally override. If you do totally override a printing message that requires forwarding, the default implementation is not invoked, which means that a vital operation might be neglected and serious errors might result.

## Using the Printing Messages

---

QuickDraw GX sends printing messages to message handlers in the message chain, including the application program, any printing extensions that are active, and the printer driver. You can override any of these messages in your printing extensions and printer drivers to modify or replace the default implementation of the message that is provided by QuickDraw GX. You can also send some of the printing messages yourself to accomplish certain tasks.

You can override printing messages to

- manage the job context
- manipulate print objects
- modify application-initiated actions
- write data to spool files
- retrieve data from spool files
- interact with users through dialog boxes
- convert spool files to universal images
- send raster image data
- send PostScript data
- send vector image data
- communicate with devices
- convert a print record in order to use QuickDraw GX to print documents originally created with QuickDraw
- display printing status messages
- add menu items to the Finder's Printing menu

The chapters “Printing Extensions” and “Printer Drivers” in this book describe how you can use the printing messages to develop your extensions and drivers. This chapter provides the reference material that you need to understand and use the messages properly.

## Printing Messages Reference

---

This section describes data types and messages that are used to develop QuickDraw GX printing extensions and printer drivers. The resources that you use to develop extensions and drivers are described in the chapter “Printing Resources” in this book.

## Printing Messages

Various sections show the structures and enumerations that are used with the printing messages.

The “Printing Messages” section describes the interface and functionality of each message that you can override in a printing extension or printer driver.

A number of the printing messages take parameters that are data structures. This section describes those data structures and the constants (enumerated types) that are used to define values within them.

The constants and data types are grouped according to usage, as follows:

- printer driver constants and data types
- Macintosh Printing Manager compatability constants and data types
- raster driver constants and data types
- PostScript driver constants and data types
- vector driver constants and data types
- user interface constants and data types

## Data Types for Printer Drivers

---

This section describes the data types that are used by printer drivers.

### The Print-to-File Structure

---

The print-to-file structure, of data type `gxPrintDestinationRec`, is used to specify information about writing a document to file. This structure defines the parameters that specify how the document is stored in a file.

```
struct gxPrintDestinationRec {
    Boolean    printToFile;
    FSSpec     fSpec;
    Boolean    includeFonts;
    Str31      fileFormat;
};

typedef struct gxPrintDestinationRec gxPrintDestinationRec,
*gxPrintDestinationPtr, **gxPrintDestinationHdl;
```

#### Field descriptions

<code>printToFile</code>	A Boolean value that is <code>true</code> if the current document is being printed to a file and <code>false</code> if not.
<code>fSpec</code>	The <code>FSSpec</code> structure for the file.
<code>includeFonts</code>	A Boolean value that is <code>true</code> if system fonts are included in the file and <code>false</code> if not. Application fonts are always included in the file.
<code>fileFormat</code>	The name of the file format to use for writing the file.

## The Printing Buffer Structure

The printing buffer structure, of data type `gxPrintingBuffer`, holds the document data that QuickDraw GX sends to the printer. It is used with the `GXDumpBuffer` and `GXFreeBuffer` messages, which are described in the section “Device Communications Messages” beginning on page 4-131.

```
struct gxPrintingBuffer {
    long    size;
    long    userData;
    char    data[1];
};

typedef struct gxPrintingBuffer gxPrintingBuffer;
```

### Field descriptions

<code>size</code>	The number of bytes in the buffer.
<code>userData</code>	The signature for this buffer, into which a message handler can write a unique ID. If you are writing a driver and overriding the <code>GXDumpBuffer</code> and <code>GXFreeBuffer</code> messages, you can use this value to match your buffers.
<code>data</code>	The data in the buffer. This is an array containing <code>size</code> bytes.

## The Page Information Structure

The page information structure, of data type `gxPageInfoRecord`, specifies information about a page that is being printed. It is used with the `GXRenderPage` message, which is described on page 4-96.

```
struct gxPageInfoRecord {
    long    docPageNum;
    long    copyNum;
    Boolean formatChanged;
    Boolean pageChanged;
    long    internalUse;
};

typedef struct gxPageInfoRecord gxPageInfoRecord;
```

### Field descriptions

<code>docPageNum</code>	The number of the page about which this structure contains information.
<code>copyNum</code>	The copy number of the page that is being printed.
<code>formatChanged</code>	A Boolean value that is <code>true</code> if this page uses a different format than the previous (most recently printed) page and <code>false</code> if not.

## Printing Messages

pageChanged	A Boolean value that is <code>true</code> if this page has different contents than the previous (most recently printed) page and <code>false</code> if not.
internalUse	A private field for internal use.

## Constants and Data Types for Macintosh Printing Manager Compatibility

---

This section describes the structure and enumerations that are used to define the universal print structure. This structure allows applications that were designed to print with the Macintosh Printing Manager to print with QuickDraw GX.

### The Universal Print Structure

---

The universal print structure, of data type `gxUniversalPrintRecord`, contains information about the current print job. You need to know about the format of this structure if you are converting information from a print record used with the Macintosh Printing Manager. This structure is used instead of the Macintosh Printing Manager `TPrint` structure. It is used with the `GXConvertPrintRecordFrom`, `GXConvertPrintRecordTo`, and `GXPrintRecordToJob` messages, which are described in the section “Compatibility Messages” beginning on page 4-147. The print record used with the Macintosh Printing Manager is described in *Inside Macintosh: Imaging With QuickDraw*.

```
struct gxUniversalPrintRecord {
    short    prVersion;

    short    appDev;
    short    appVRes;
    short    appHRes;
    Rect     appPage;
    Rect     appPaper;

    short    devType;
    short    pageV;
    short    pageH;
    char     filler;
    char     feed;

    short    devKind;
    short    devVRes;
    short    devHRes;
    Rect     devPage;

    short    actualCopies;
    short    options;
    short    reduction;
```

## Printing Messages

```

        char        orientation;

        char        qualityMode;
        char        firstTray;
        char        remainingTray;

        char        coverPage;
        char        headMotion;
        char        saveFile;

        char        userCluster1;
        char        userCluster2;
        char        userCluster3;

        short       firstPage;
        short       lastPage;
        short       copies;
        char        reserved1;
        char        reserved2;
        PrIdleProcPtr pIdleProc;
        Ptr         pFileName;
        short       fileVol;
        char        fileVers;
        char        reserved3;

        short       printX[19];
    };

typedef struct gxUniversalPrintRecord gxUniversalPrintRecord,
*gxUniversalPrintRecordPtr, **gxUniversalPrintRecordHdl;

```

**Field descriptions**

<code>prVersion</code>	The version ID of this converted print record. This value is always the constant <code>gxPrintRecordVersion</code> , a value of 8.
<code>appDev</code>	The kind of device this print record applies to. At this time, this field always has the value 0.
<code>appVRes</code>	The vertical resolution of the application.
<code>appHres</code>	The horizontal resolution of the application.
<code>appPage</code>	The page size for this print record, stated in terms of the application resolution.
<code>appPaper</code>	The paper rectangle for this print record. This value is an offset from the value of the <code>appPage</code> field (the page always fits within the paper, and the edge of the paper is a negative offset from the page).
<code>devType</code>	The device type for this print record. At this time, the value of this field is always 0.

## Printing Messages

pageV	The vertical page height in 120ths of an inch.
pageH	The horizontal page width in 120ths of an inch.
filler	Unused.
feed	The feed mode used for this print record. The possible values are given in the section “Feed Mode Options for the Universal Print Structure” on page 4-15.
devKind	The kind of device for this print record. At this time, the value of this field is always 0xA900.
devVRes	The vertical resolution of the device.
devHRes	The horizontal resolution of the device.
devPage	The page size of the device.
actualCopies	The actual number of copies for this print job.
options	The options for this device. The constants that you can combine into a single value for this field are given in the “Print Options for the Universal Print Structure” on page 4-15.
reduction	The reduction or enlargement factor used for this print job.
orientation	The orientation of the paper. The possible values are given in the section “Paper Orientation Options for the Universal Print Structure” on page 4-16.
qualityMode	The quality mode used for this print job. The possible values are given in the section “Print Quality Modes for the Universal Print Structure” on page 4-17.
firstTray	Which of the printer’s paper trays is used as the first paper tray for this print job. The possible values are given in the section “Paper Tray Selections for the Universal Print Structure” on page 4-17.
remainingTray	The other paper tray used for this print job. The possible values are given in the section “Paper Tray Selections for the Universal Print Structure” on page 4-17.
coverPage	The type of cover page to print for this print job. The possible values are given in the section “Cover Page Options for the Universal Print Structure” on page 4-18.
headMotion	The type of print-head motion to use for this print job. The possible values are given in the section “Print-Head Motions for the Universal Print Structure” on page 4-18.
saveFile	The type of file to save for this print job. The possible values are given in the section “File Save Types for the Universal Print Structure” on page 4-19. This field applies only to PostScript print jobs.
userCluster1	Unused.
userCluster2	Unused.
userCluster3	Unused.
firstPage	The number of the first page.
lastPage	The number of the last page.
copies	The number of copies. This value is always 1.
reserved1	Unused.



## Printing Messages

<code>reserved2</code>	Unused.
<code>pIdleProc</code>	A pointer to the idle procedure to use with this print job.
<code>pFileName</code>	A pointer to the name of the spool file for this print job.
<code>fileVol</code>	The volume reference number of the spool file volume.
<code>fileVers</code>	The spool file version, which must be 0.
<code>reserved3</code>	Reserved. This field must have a value of 0.
<code>printX</code>	Reserved for internal use.

## Feed Mode Options for the Universal Print Structure

---

You can use the feed mode constants to define the value of the `feed` field in the universal print structure.

```
enum {
    gxAutoFeed      = 0,
    gxManualFeed    = 1
};
```

### Constant descriptions

<code>gxAutoFeed</code>	The print job uses automatic paper feeding.
<code>gxManualFeed</code>	The print job uses manual paper feeding.

## Print Options for the Universal Print Structure

---

You can use the print option constants to define the value of the `options` field in the universal print structure. You can combine these constants into a single value.

```
enum {
    gxPreciseBitmap      = 0x0001,
    gxBiggerPages        = 0x0002,
    gxGraphicSmoothing   = 0x0004,
    gxTextSmoothing      = 0x0008,
    gxFontSubstitution    = 0x0010,
    gxInvertPage         = 0x0020,
    gxFlipPageHoriz      = 0x0040,
    gxFlipPageVert       = 0x0080,
    gxColorMode          = 0x0100,
    gxBidirectional      = 0x0200,
    gxUserFlag0          = 0x0400,
    gxUserFlag1          = 0x0800,
    gxUserFlag2          = 0x1000,
    gxReservedFlag0      = 0x2000,
    gxReservedFlag1      = 0x4000,
    gxReservedFlag2      = 0x8000
};
```

## Printing Messages

**Constant descriptions**

<code>gxPreciseBitmap</code>	The driver needs to format pages as tall-adjusted for the Apple ImageWriter family of printers and to use precise bitmaps for the Apple LaserWriter family of printers.
<code>gxBiggerPages</code>	The driver needs to not apply gaps if printing this job to one of the Apple ImageWriter family of printers, and the driver needs to use a large print area for the Apple LaserWriter family of printers.
<code>gxGraphicSmoothing</code>	The driver needs to perform graphics smoothing on the Apple LaserWriter family of printers.
<code>gxTextSmoothing</code>	The driver needs to perform text smoothing for this print job.
<code>gxFontSubstitution</code>	The driver needs to perform font substitution for this print job.
<code>gxInvertPage</code>	The driver needs to invert the printed image (convert white to black and black to white) for this print job.
<code>gxFlipPageHoriz</code>	The driver needs to flip pages horizontally for this print job.
<code>gxFlipPageVert</code>	The driver needs to flip pages vertically for this print job.
<code>gxColorMode</code>	This print job uses color printing.
<code>gxBidirectional</code>	This print job uses bidirectional printing.

## Paper Orientation Options for the Universal Print Structure

---

You can use the paper orientation option constants to define the value of the orientation field in the universal print structure.

```
enum {
    gxPortraitOrientation      = 0,
    gxLandscapeOrientation     = 1,
    gxAltPortraitOrientation   = 2,
    gxAltLandscapeOrientation  = 3
};
```

**Constant descriptions**

<code>gxPortraitOrientation</code>	The driver needs to print the print job in portrait orientation.
<code>gxLandscapeOrientation</code>	The driver needs to print the print job in landscape orientation.

## Printing Messages

`gxAltPortraitOrientation`

The driver needs to print the print job in rotated (90 degrees) portrait orientation.

`gxAltLandscapeOrientation`

The driver needs to print the print job in rotated (90 degrees) landscape orientation.

## Print Quality Modes for the Universal Print Structure

---

You can use the print quality mode constants to define the value of the `qualityMode` field in the universal print structure.

```
enum {
    gxBestQuality          = 0,
    gxFasterQuality        = 1,
    gxDraftQuality         = 2
};
```

### Constant descriptions

`gxBestQuality` The driver needs to print this print job with highest-quality printing.

`gxFasterQuality`

The driver needs to print this print job with medium-quality printing.

`gxDraftQuality` The driver needs to print this print job with fastest-quality printing.

## Paper Tray Selections for the Universal Print Structure

---

You can use the paper tray selection constants to define the value of the `firstTray` and `remainingTray` fields in the universal print structure.

```
enum {
    gxFirstTray    = 0,
    gxSecondTray   = 1,
    gxThirdTray    = 2
};
```

### Constant descriptions

`gxFirstTray` The printer's first paper tray is used for this print job.

`gxSecondTray` The printer's second paper tray is used for this print job.

`gxThirdTray` The printer's third paper tray is used for this print job.

## Cover Page Options for the Universal Print Structure

---

You can use the cover page option constants to define the value of the `coverPage` field in the universal print structure.

```
enum {
    gxNoCoverPage      = 0,
    gxFirstPageCover   = 1,
    gxLastPageCover    = 2
};
```

### Constant descriptions

`gxNoCoverPage`    The driver need not print a cover page for this print job.

`gxFirstPageCover`    The driver needs to print a cover page before the first page for this print job.

`gxLastPageCover`    The driver needs to print a cover page after the last page for this print job.

## Print-Head Motions for the Universal Print Structure

---

You can use the print-head motion constants to define the value of the `headMotion` field in the universal print structure.

```
enum {
    gxUnidirectionalMotion = 0,
    gxBidirectionalMotion  = 1
};
```

### Constant descriptions

`gxUnidirectionalMotion`    The driver needs to move the print head in only one direction for this print job.

`gxBidirectionalMotion`    The driver needs to move the print head in both directions for this print job.

## File Save Types for the Universal Print Structure

---

You can use the constants that define how to save a print file to define the value of the `saveFile` field in the universal print structure.

```
enum {
    gxNoFile          = 0,
    gxPostScriptFile  = 1
};
```

### Constant descriptions

`gxNoFile`                The driver need not save the print job in a file.

`gxPostScriptFile`       The driver needs to save the print job in a PostScript file.

## Constants and Data Types for the Raster Imaging System

---

This section describes the enumerations and structures that are used by the raster imaging system messages. You need to know about these data types if you are writing a printing extension or printer driver for a printing device that uses the raster imaging system.

### Raster Offscreen Structure

---

The raster offscreen structure, of type `gxOffscreenRec`, contains an offscreen area for a bitmap.

```
struct gxOffscreenRec {
    short      numberOfPlanes;
    Handle      offscreenStorage;
    gxOffscreenPlaneRec
                thePlanes[1];
};

typedef struct gxOffscreenRec gxOffscreenRec, *gxOffscreenPtr,
**gxOffscreenHdl;
```

### Field descriptions

`numberOfPlanes`       The number of planes into which to draw the bitmap.

`offScreenStorage`       A handle to the bitmap's image data.

`thePlanes`            An array of plane structures into which to draw the data. Each element of this array is an offscreen plane structure, which is described in the next section.

## Raster Offscreen Plane Structure

---

The raster offscreen plane structure, of data type `gxOffscreenPlaneRec`, defines the format of one plane in the offscreen planar area.

```
struct gxOffscreenPlaneRec {
    gxViewPort      theViewPort;
    gxViewDevice     theDevice;
    gxViewGroup      theViewGroup;
    gxShape           theBitmap;
    gxBitMap          theBits;
};

typedef struct gxOffscreenPlaneRec gxOffscreenPlaneRec;
```

### Field descriptions

<code>theViewPort</code>	The view port for this offscreen plane.
<code>theDevice</code>	The view device for this offscreen plane.
<code>theViewGroup</code>	The view group for this offscreen plane.
<code>theBitmap</code>	The shape object that represents the offscreen bitmap.
<code>theBits</code>	The actual bitmap data for this area.

## Raster Offscreen Setup Structure

---

The raster offscreen setup structure, of data type `gxOffscreenSetupRec`, defines how to store raster data in offscreen memory.

```
struct gxOffscreenSetupRec {
    short      width;
    short      minHeight;
    short      maxHeight;
    Fixed       ramPercentage;
    long       ramSlop;
    short      depth;
    gxMapping   vpMapping;
    gxMapping   vdMapping;
    short      planes;
    gxPlaneSetupRec
                planeSetup[4];
};

typedef struct gxOffscreenSetupRec gxOffscreenSetupRec;
```

## Printing Messages

**Field descriptions**

<code>width</code>	The width in pixels of the raster.
<code>minHeight</code>	The minimum height in pixels. The actual height that was created for the offscreen storage is stored into this value.
<code>maxHeight</code>	The maximum height in pixels.
<code>ramPercentage</code>	The maximum percentage of available RAM to use for this storage.
<code>ramSlop</code>	The amount of RAM that must be left available after allocating for this storage.
<code>depth</code>	The depth, in bits, of each plane.
<code>vpMapping</code>	The mapping for offscreen view ports.
<code>vdMapping</code>	The mapping for offscreen view devices.
<code>planes</code>	The number of planes to allocate, each of which will have depth bits allocated for it.
<code>planeSetup</code>	The parameters of each plane, defined in the <code>gxPlaneSetupRec</code> structure, which is described in the next section. Four of these records are allocated because most drivers use four planes.

**Raster Offscreen Plane Setup Structure**

The raster offscreen plane setup structure, of data type `gxPlaneSetupRec`, defines settings for storing a single plane of raster data in offscreen memory. The raster offscreen setup structure, described in the previous section, uses four of these structures.

```
struct gxPlaneSetupRec {
    gxRasterPlaneOptions    planeOptions;
    gxHalftone              planeHalftone;
    gxColorSpace            planeSpace;
    gxColorSet              planeSet;
    gxColorProfile          planeProfile;
};

typedef struct gxPlaneSetupRec gxPlaneSetupRec;
```

**Field descriptions**

<code>planeOptions</code>	The options for this plane. This value is the combined values of the constants that you include from the raster plane options enumeration, which is described in the next section.
<code>planeHalftone</code>	The halftone structure for this plane. This is optional.
<code>planeSpace</code>	The color space for this plane. The color-space value constants are shown in the chapter “Colors and Color-Related Objects” in <i>Inside Macintosh: QuickDraw GX Objects</i> . You can specify the value <code>gxNoSpace</code> to use the default color space.

## Printing Messages

<code>planeSet</code>	The color set for this plane. You can specify <code>nil</code> to use the default color set.
<code>planeProfile</code>	The color profile for this plane. You can specify <code>nil</code> if you do not want color matching applied to this plane.

## Raster Plane Options

---

The raster plane options enumeration defines constants that you can use in the `planeOptions` field of the raster offscreen plane setup structure.

```
enum {
    gxDefaultOffscreen      = 0x00000000,
    gxDontSetHalftone       = 0x00000001,
    gxDotTypeIsDitherLevel  = 0x00000002
};
```

```
typedef long gxRasterPlaneOptions;
```

### Constant descriptions

`gxDefaultOffscreen`

The driver allocates bits for the print job and creates halftone values. This is the default value.

`gxDontSetHalftone`

The driver does not call the `GXSetViewPortHalftone` function.

`gxDotTypeIsDitherLevel`

The driver calls the `GXSetViewPortDither` function and uses the `gxDotType` constant as the dithering level.

## Raster Package Bitmap Structure

---

The raster package bitmap structure, of data type `gxRasterPackageBitmapRec`, is used with the `GXRasterPackageBitmap` message.

```
struct gxRasterPackageBitmapRec {
    gxBitmap      *bitmapToPackage;
    unsigned short startRaster;
    unsigned short colorBand;
    Boolean       isBandDirty;
    Rect          dirtyRect;
};
```

```
typedef struct gxRasterPackageBitmapRec gxRasterPackageBitmapRec;
```



## Printing Messages

**Field descriptions**

<code>bitmapToPackage</code>	A pointer to the bitmap that contains the data to be packaged.
<code>startRaster</code>	The raster where the packaging is to begin.
<code>colorBand</code>	Which color pass of packaging this structure represents.
<code>isBandDirty</code>	A Boolean value that is <code>true</code> if the raster band being packaged contains any nonwhite bits and is <code>false</code> if the raster band is all white.
<code>dirtyRect</code>	A rectangle that defines the area in the bitmap that contains dirty bits.

**Raster Imaging System Structure**

The raster imaging system structure, of data type `gxRasterImageDataRec`, is the structure that QuickDraw GX uses to maintain information about the state of the raster imaging system. This structure is used with the `GXRasterLineFeed` and `GXRasterPackageBitmap` messages, which are described in the section “Raster Imaging Messages” beginning on page 4-97.

```
struct gxRasterImageDataRec {
    gxRasterRenderOptions    renderOptions;
    Fixed                    hImageRes;
    Fixed                    vImageRes;
    short                    minBandSize;
    short                    maxBandSize;
    gxRectangle              pageSize;
    short                    currentYPos;
    gxRasterPackageRec       packagingInfo;
    Boolean                  optionsValid;
    gxRasterPackageControlsRec packageControls;
    gxOffscreenSetupRec      theSetup;
};
```

```
typedef struct gxRasterImageDataRec gxRasterImageDataRec,
*gxRasterImageDataPtr, **gxRasterImageDataHdl;
```

**Field descriptions**

<code>renderOptions</code>	Rendering options for raster imaging. This value is the combined values of the constants that you include from the raster render options enumeration, which is described in the next section.
<code>hImageRes</code>	The horizontal resolution for imaging.
<code>vImageRes</code>	The vertical resolution for imaging.
<code>minBandSize</code>	The minimum band size to use, in pixels.
<code>maxBandSize</code>	The maximum band size to use, in pixels.
<code>pageSize</code>	The size of the page in pixels.

## Printing Messages

<code>currentYPos</code>	The current position on the page.
<code>packagingInfo</code>	The raster package structure.
<code>optionsValid</code>	A Boolean value that is <code>true</code> if options are specified for the packaging messages and <code>false</code> if not.
<code>packageControls</code>	The raster package controls structure.
<code>theSetup</code>	The setup for the offscreen planar area data. This is a variable length object.

## Raster Render Options

---

The raster render options enumeration defines constants that you can use in the `renderOptions` field of the raster imaging system structure, which is described in the previous section.

```
enum {
    gxDefaultRaster                = 0x00000000,
    gxDontResolveTransferModes     = 0x00000001,
    gxRenderInReverse              = 0x00000002,
    gxOnePlaneAtATime              = 0x00000004,
    gxSendAllBands                 = 0x00000008
};
```

```
typedef long gxRasterRenderOptions;
```

### Constant descriptions

<code>gxDefaultRaster</code>	The driver uses the default raster options.
<code>gxDontResolveTransferModes</code>	If this is included, the driver does not need to resolve transfer modes. The default is to resolve transfer modes.
<code>gxRenderInReverse</code>	The driver needs to traverse the raster image in reverse order.
<code>gxOnePlaneAtATime</code>	The driver needs to render each plane separately.
<code>gxSendAllBands</code>	The driver needs to send every band of data, even if it is all empty.

## Raster Package Structure

---

The raster package structure, of data type `gxRasterPackageRec`, is used to define how raster data is packaged into a buffer for sending to the output device.

```
struct gxRasterPackageRec {
    Ptr        bufferSize;
    short      colorPasses;
```

Printing Messages

```
        short                headHeight;
        short                numberPasses;
        short                passOffset;
        gxRasterPackageOptions  packageOptions;
};

typedef struct gxRasterPackageRec gxRasterPackageRec,
*gxrasterPackagePtr, **gxRasterPackageHdl;
```

Field descriptions

bufferSize	The number of bytes in the raster packaging buffer.
colorPasses	The number of print-head passes required to print all colors.
headHeight	The height of the print head in pixels.
numberPasses	The number of passes required per headheight.
passOffset	The offset between passes in pixels.
packageOptions	The packaging options, as defined in the next section.

Raster Package Options

The raster package options enumeration defines constants that you can use in the packageOptions field of the raster package structure, as described in the previous section.

```
enum {
    gxSendAllColors    = 0x00000001,
    gxInterlaceColor   = 0x00000002,
    gxOverlayColor     = 0x00000004,
    gxUseColor         = (gxInterlaceColor|gxOverlayColor)
};

typedef long gxRasterPackageOptions;
```

Constant descriptions

gxSendAllColors	The driver needs to send all bands of data in the raster package, including bands that are all white.
gxInterlaceColor	The driver needs to interlace color data to prevent the ribbon from becoming contaminated.
gxOverlayColor	The driver can send colors without interlacing because the output device doesn't have a ribbon contamination problem.
gxUseColor	The driver needs to send color data. This option implies either gxInterlaceColor or gxOverlayColor.

## Constants and Data Types for the PostScript Imaging System

---

This section describes the enumerations and structures that are used by the PostScript imaging system messages. You need to know about these data types if you are writing a printing extension or printer driver for a printing device that uses the PostScript imaging system.

### PostScript Imaging System Structure

---

The PostScript imaging system structure, of data type `gxPostScriptImageDataRec`, is allocated and set up by QuickDraw GX during the implementation of the `GXPostScriptGetDocumentProcSetList` and `GXPostScriptDownloadProcSetList` messages, whose descriptions begin on page 4-112. This structure contains information describing the contents of a document that is to be imaged and quantifies the limitations of the device to which the document is to be imaged. QuickDraw GX fills in all the fields in the structure with default values, but your printing extension or printer driver can change any of these fields or use them to derive information about the imaging of the document. For more information about how the fields in this structure are used for color printing, see the section “Color Printing” beginning on page 3-50 in the chapter “Printer Drivers.”

```
struct gxPostScriptImageDataRec {
    short                languageLevel;
    gxColorSpace         devCSpace;
    gxColorProfile       devCProfile;
    gxPostScriptRenderOptions renderOptions;
    long                pathLimit;
    short               gsaveLimit;
    short               opStackLimit;
    scalerStreamTypeFlag fontType;
    long               printerVM;
    long               reserved0;
};

typedef struct gxPostScriptImageDataRec gxPostScriptImageDataRec,
*gxPostScriptImageDataPtr, **gxPostScriptImageDataHandle;
```

#### Field descriptions

<code>languageLevel</code>	The language level for the PostScript printer to which your driver is imaging the document.
<code>devCSpace</code>	The color space used for this device. The color-space value constants are shown in the chapter “Colors and Color-Related Objects” in <i>Inside Macintosh: QuickDraw GX Objects</i> . The values that are valid for this field are <code>gxGraySpace</code> , <code>gxCMYKSpace</code> , and <code>gxRGBSpace</code> .

## Printing Messages

<code>devCProfile</code>	The color profile for this device. Color profiles are explained in the chapter “Colors and Color-Related Objects” in <i>Inside Macintosh: QuickDraw GX Objects</i> .
<code>renderOptions</code>	Rendering options for the PostScript printer. This value is the combined value of the constants that you include from the PostScript render options enumeration, which is described in the next section.
<code>pathLimit</code>	The largest number of points that can be active in the device during the imaging process without generating a PostScript <code>limitcheck</code> error, which is described in <i>PostScript Language Reference Manual</i> , 2nd Edition.
<code>gsaveLimit</code>	The greatest number of <code>gsaves</code> that can be performed on the device without generating a PostScript <code>limitcheck</code> error.
<code>opStackLimit</code>	The maximum number of objects that can be placed in the stack at any one time without generating a PostScript <code>limitcheck</code> error.
<code>fontType</code>	The stream type that the imaging system needs to use when downloading a font. The possible values are described in <i>Inside Macintosh: Typography</i> .
<code>printerVM</code>	The amount of memory available in the printer.

## PostScript Render Options

The PostScript render options enumeration defines constants for use in the `renderOptions` field of the PostScript imaging system structure, which is described in the previous section.

```
enum
{
    gxNeedsAsciiOption          = 0x00000001,
    gxNeedsCommentsOption       = 0x00000002,
    gxBoundingBoxesOption       = 0x00000004,
    gxPortablePostScriptOption   = 0x00000008,
    gxUseLevel2ColorOption      = 0x00000080
};
```

```
typedef long gxPostScriptRenderOptions;
```

### Constant descriptions

`gxNeedsAsciiOption`

The driver needs to convert all binary data to 7-bit ASCII values.

`gxNeedsCommentsOption`

The driver needs to issue PostScript comments.

`gxBoundingBoxesOption`

The driver needs to calculate values for the `%%BoundingBox:` and `%%PageBoundingBox:` variables. This option implies the `gxNeedsCommentsOption` constant.

## Printing Messages

`gxPortablePostScriptOption`

The driver needs to generate PostScript that is not device specific.

`gxUseLevel2ColorOption`

The driver uses Level 2 device-independent color when printing to a Level 2 output device.

## PostScript Glyphs Structure

---

QuickDraw GX constructs the PostScript glyphs structure, of data type `gxPrinterGlyphsRec`, so a driver can communicate with the imaging system about the fonts and glyphs that are available on a printer. This structure is used by the `GXPostScriptGetPrinterGlyphsInformation` message, which is described on page 4-116.

```
struct gxPrinterGlyphsRec {
    gxFont          theFont;
    long            nGlyphs;
    gxFontPlatform  platform;
    gxFontScript    script;
    gxFontLanguage  language;
    long            vmUsage;
    unsigned long    glyphBits[1];
};

typedef struct gxPrinterGlyphsRec gxPrinterGlyphsRec;
```

### Field descriptions

<code>theFont</code>	The font that this information is for.
<code>nGlyphs</code>	The number of glyphs that the font contains.
<code>platform</code>	The platform of the font.
<code>script</code>	The script code of the font. This value has no meaning if the value of the <code>platform</code> field is -1.
<code>language</code>	The language code of the font. This value has no meaning if the value of the <code>platform</code> field is -1.
<code>vmUsage</code>	The amount of printer memory required for the font.
<code>glyphBits</code>	A variable-sized array containing the data for the glyphs. This array must be aligned on a long word boundary. Its size in long words can be computed by the formula: $(nGlyphs + 31) / 32$ .

You can read about font glyphs, platforms, scripts, and languages in *Inside Macintosh: QuickDraw GX Typography*.

## PostScript Procedure Set List Structure

The PostScript procedure set list structure, of data type `gxProcSetListRec`, contains a list of the procedures needed to image the specified document. This structure is used by the `GXPostScriptGetDocumentProcSetList` and `GXPostScriptDownloadProcSetList` messages, whose descriptions begin on page 4-112.

```
struct gxProcSetListRec {
    Signature    clientId;
    OSType       controlType;
    short        controlId;
    OSType       dataType;
    long         reserved0;
};

typedef struct gxProcSetListRec gxProcSetListRec,
*gxProcSetListPtr, **gxProcSetListHdl;
```

### Field descriptions

<code>clientId</code>	The unique ID provided by the printing extension or printer driver to avoid resource conflicts.
<code>controlType</code>	A resource type that serves as a control resource for the procedure set.
<code>controlId</code>	The ID of the resource that controls the downloading of the procedure set.
<code>dataType</code>	The resource type that, in combination with the control resource, specifies which resources are to be interpreted as belonging to the procedure set.

## PostScript Query Results

The PostScript query results enumeration defines constants that are the possible results of the `GXPostScriptQueryPrinter` message, which is described on page 4-101.

```
enum {
    gxPrinterOK           = 0,
    gxInitializePrinter    = 1,
    gxFilePrinting        = 2,
    gxResetPrinter        = 128
};
```

## Printing Messages

**Constant descriptions**

<code>gxPrinterOK</code>	The printer responded to the query with enough information to initiate printing of the job.
<code>gxInitializePrinter</code>	The printer responded to the query by indicating that it needs to be initialized.
<code>gxFilePrinting</code>	The printer responded to the query by indicating that the print job has been redirected to a file.
<code>gxResetPrinter</code>	The printer responded to the query by indicating that it needs to be restarted.

## Constants and Data Types for the Vector Imaging System

---

This section describes the enumerations and structures that are used with the vector imaging system messages. You need to know about these data types if you are writing a printing extension or printer driver for a printing device that uses the vector imaging system.

### Vector Halftone Structure

---

The vector halftone structure, of data type `gxVHalftoneRec`, defines the halftone information for a vector device. This structure is used with the vector imaging system structure, which is described on page 4-32.

```
struct gxVHalftoneRec {
    gxColorSpace      halftoneSpace;
    gxHalftoneCompRec halftoneComps[4];
    long              penIndexForBW;
};

typedef struct gxVHalftoneRec gxVHalftoneRec;
```

**Field descriptions**

<code>halftoneSpace</code>	The color space for this device. The color-space value constants are shown in the chapter “Colors and Color-Related Objects” in <i>Inside Macintosh: QuickDraw GX Objects</i> .
<code>halftoneComps</code>	An array of vector halftone component structures. This array has four entries, one for each color component. The vector halftone component structure is described in the next section.
<code>penIndexForBw</code>	The pen index for drawing 1-bit deep or black-and-white bitmaps.

### Vector Halftone Component Structure

---

The vector halftone component structure, of data type `gxVHalftoneCompRec`, describes the values to use for creating a halftone for a color component.



## Printing Messages

```

struct gxVHalftoneCompRec {
    Fixed    angle;
    long     penIndex;
};

typedef struct gxVHalftoneCompRec gxVHalftoneCompRec;

```

**Field descriptions**

angle	The halftone angle for this component.
penIndex	The index of the pen to use for drawing this component.

**Vector Shape Structure**

The vector shape structure, of data type `gxVectorShapeDataRec`, is used with the `GXVectorVectorizeShape` message, which is described on page 4-130.

```

struct gxVectorShapeDataRec {
    gxVectorShapeOptions    shapeOptions;
    long                    maxPolyPoints;
    Fixed                   shapeError;
    Fixed                   textSize;
    Fixed                   frameSize;
};

typedef struct gxVectorShapeDataRec gxVectorShapeDataRec;

```

**Field descriptions**

shapeOptions	Options that you can include for rendering shapes into vectors. This value is the combined value of the constants that you include from the vector shape options enumeration, which is described in the next section.
maxPolyPoints	The maximum number of points that a polygon can contain for drawing on this device. If the value of this field is 0, QuickDraw GX converts polygons into vectors rather than sending polygons to the device.
shapeError	The maximum number of pixels by which the vectors created to approximate a curve can deviate from the original. The smaller this value, the more vectors are used to approximate each curve, which means more memory, but less deviation in shape. A value of 0x00002000 is typically used, which allows an error of no more than one 1/8 of a pixel at 72 dots per inch.
textSize	Any text above this size is filled, while any text below this size is outlined but not filled.
frameSize	Any shapes with frames larger than this size are filled, while shapes with any shapes with frames smaller than this size are stroked.

## Vector Shape Options

---

The vector shape options enumeration provides constants that you can use in the `shapeOptions` field of the vector shape structure, which is described in the previous section.

```
enum {
    gxUnidirectionalFill      = 0x00000001,
    gxAlsoOutlineFilledShape = 0x00000002
};

typedef long gxVectorShapeOptions;
```

### Constant descriptions

`gxUnidirectionalFill`

The driver needs to generate scan lines in one direction only. This value is useful for transparencies.

`gxAlsoOutlineFilledShapes`

The driver also needs to draw the outlines of filled shapes.

## Vector Imaging System Structure

---

The vector imaging system structure, of data type `gxVectorImageDataRec`, is the structure that QuickDraw GX uses to maintain information about the state of the vector imaging system. This data type is used with the `GXRenderPage` message, which is described on page 4-130.

```
struct gxVectorImageDataRec {
    gxVectorRenderOptions  renderOptions;
    Fixed                  devRes;
    gxTransform             devTransform;
    gxColorSet              clrSet;
    gxColor                 bgColor;
    gxVHaltoneRec           halftoneInfo;
    gxPenTableHdl           hPenTable;
    gxRectangle             pageRect;
    gxVectorShapeDataRec    shapeData;
};

typedef struct gxVectorImageDataRec gxVectorImageDataRec,
*gxVectorImageDataPtr, **gxVectorImageDataHdl;
```

### Field descriptions

`renderOptions` Options to control the vector rendering. This value is the combined values of the constants that you include from the vector render options enumeration, which is described in the next section.

`devRes` The resolution of the device.

## Printing Messages

<code>devTransform</code>	The transform for the device.
<code>clrSet</code>	The entire set of colors available on the device.
<code>bgColor</code>	The background color in the specified color space.
<code>halftoneInfo</code>	The halftone information for color bitmaps. This structure is described in the section “Vector Halftone Structure” on page 4-30.
<code>hPenTable</code>	The complete list of pens available on the device, along with the thickness and pen position of each. This structure is described in the section “Vector Pen Table Structure” on page 4-34.
<code>pageRect</code>	The dimensions of the page. If you set the coordinates of this rectangle to all zeros, the page format values are used.
<code>shapeData</code>	The information on how to render a shape on the device.

## Vector Render Options

---

The vector render options enumeration defines constants for use in the `renderOptions` field of the vector imaging system structure.

```
enum {
    gxColorSort          = 0x00000001,
    gxATransferMode      = 0x00000002,
    gxNoOverlap          = 0x00000004,
    gxAColorBitmap       = 0x00000008,
    gxSortbyPenPos       = 0x00000010,
    gxPenLessPlotter     = 0x00000020,
    gxCutterPlotter      = 0x00000040,
    gxNoBackGround       = 0x00000080
};

typedef long gxVectorRenderOptions;
```

### Constant descriptions

<code>gxColorSort</code>	You need to specify this option for pen plotters.
<code>gxATransferMode</code>	The driver needs to resolve transfer modes.
<code>gxNoOverlap</code>	The driver needs to produce non-overlapping output.
<code>gxAColorBitmap</code>	The driver needs to produce color bitmap output.
<code>gxSortbyPenPos</code>	The driver needs to draw shapes in the order of the pens in the pen table (which is not the same as the order of the pens in the pen carousel).
<code>gxPenLessPlotter</code>	The output device is a raster printer or plotter.
<code>gxCutterPlotter</code>	The output device has a cutter.
<code>gxNoBackGround</code>	Shapes that map to the background color are not sent to the driver.

## Vector Pen Table Structure

---

The vector pen table structure, of data type `gxPenTable`, defines the pens that are available for a pen table on a vector device. This structure is used with the vector imaging system structure, which is described on page 4-32, and with the `GXVectorLoadPens` message, which is described on page 4-128.

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

typedef struct gxPenTable gxPenTable, *gxPenTablePtr,
**gxPenTableHdl;
```

### Field descriptions

<code>numPens</code>	The number of pen table entry structures in the <code>pens</code> array.
<code>pens</code>	An array of pen table entry structures, with one entry for each pen that is available in this pen table.

## Vector Pen Table Entry Structure

---

The vector pen table entry structure, of data type `gxPenTableEntry`, defines a single pen that is in a pen table used on a vector device.

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

typedef struct gxPenTableEntry gxPenTableEntry;
```

### Field descriptions

<code>penName</code>	The name of the pen that this structure describes.
<code>penColor</code>	The color of this pen. This color matches one of the entries in the color set for this device.
<code>penThickness</code>	The size of this pen.
<code>penUnits</code>	The units that define the <code>penThickness</code> value, as described in the next section.
<code>penPosition</code>	The position of this pen in the devices's pen carousel. If the pen is not loaded, it is set to the value <code>kPenNotLoaded</code> (-1).

## Vector Pen Units

The vector pen units enumeration defines the kind of units used to specify the thickness of a pen table entry, as described in the previous section.

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

### Constant descriptions

<code>gxDeviceUnits</code>	The pen thickness is specified in device units.
<code>gxMMUnits</code>	The pen thickness is specified in millimeters.
<code>gxInchesUnits</code>	The pen thickness is specified in inches.

## User Interface Constants and Data Types

This section describes the enumerations and structures that QuickDraw GX uses with the dialog box messages. You need to know about these data types when writing the user interface portion of a printing extension or printer driver.

### The Panel Information Structure

The panel information structure, of data type `gxPanelInfoRecord`, provides information to the panel about the current dialog box and panel event. This structure is used with the `GXHandlePanelEvent` and `GXFilterPanelEvent` messages, whose descriptions begin on page 4-85.

```
struct gxPanelInfoRecord {
    gxPanelEvent    panelEvt;
    short           panelResId;
    DialogPtr       pDlg;
    EventRecord     *theEvent;
    short           itemHit;
    short           itemCount;
    short           evtAction;
    short           errorStringId;
    gxFormat        theFormat;
    void            *refCon;
};

typedef struct gxPanelInfoRecord gxPanelInfoRecord;
```

## Printing Messages

**Field descriptions**

<code>panelEvt</code>	The event to filter or handle.
<code>panelResId</code>	The resource ID of the current panel ('ppnl') resource.
<code>pDlg</code>	A pointer to the dialog box structure.
<code>theEvent</code>	A pointer to the event that occurred.
<code>itemHit</code>	The actual item number where the event occurred, using the item-numbering scheme of the Dialog Manager.
<code>itemCount</code>	The item count before your panel's items in the dialog box.
<code>evtAction</code>	The action that results once this event is processed. This value is one of the constants defined in the panel event actions enumeration, which is described on page 4-38. This field is only meaningful for filtering, and is used for parsing.
<code>errorStringId</code>	The ID of the 'STR' resource to put in the error alert. A value of 0 in this field indicates that there is no error string to display.
<code>theFormat</code>	The current format. This is only meaningful in a Custom Page Setup dialog box.
<code>refcon</code>	A reference constant for use by the generator of the panel.

**Panel Events**


---

The panel event enumeration defines the possible event types that can occur in a panel. This data type is used with the panel information structure, which is described in the previous section.

```
enum {
    gxPanelNoEvt          = (gxPanelEvent) 0,
    gxPanelOpenEvt        = (gxPanelEvent) 1,
    gxPanelCloseEvt       = (gxPanelEvent) 2,
    gxPanelHitEvt         = (gxPanelEvent) 3,
    gxPanelActivateEvt    = (gxPanelEvent) 4,
    gxPanelDeactivateEvt  = (gxPanelEvent) 5,
    gxPanelIconFocusEvt   = (gxPanelEvent) 6,
    gxPanelPanelFocusEvt  = (gxPanelEvent) 7,
    gxPanelFilterEvt      = (gxPanelEvent) 8,
    gxPanelCancelEvt      = (gxPanelEvent) 9,
    gxPanelConfirmEvt     = (gxPanelEvent) 10,
    gxPanelDialogEvt      = (gxPanelEvent) 11,
    gxPanelOtherEvt       = (gxPanelEvent) 12,
    gxUserWillConfirmEvt  = (gxPanelEvent) 13
};

typedef long gxPanelEvent;
```

## Printing Messages

**Constant descriptions**

<code>gxPanelNoEvt</code>	No event has occurred.
<code>gxPanelOpenEvt</code>	The panel is about to open. It needs to be initialized and drawn.
<code>gxPanelCloseEvt</code>	The panel is about to close.
<code>gxPanelHitEvt</code>	The user has selected an item in the panel.
<code>gxPanelActivateEvt</code>	The dialog box in which the panel resides has just been activated.
<code>gxPanelDeactivateEvt</code>	The dialog box in which the panel resides is about to be deactivated.
<code>gxPanelIconFocusEvt</code>	The focus has changed from the panel to the icon list.
<code>gxPanelPanelFocusEvt</code>	The focus has changed from the icon list to the panel.
<code>gxPanelFilterEvt</code>	The panel event needs to be filtered.
<code>gxPanelCancelEvt</code>	The user has selected the Cancel button in the dialog box.
<code>gxPanelConfirmEvt</code>	The user has selected the OK button in the dialog box.
<code>gxPanelDialogEvt</code>	An event has occurred in the panel that is going to be handled by a dialog box handler such as the application, a printing extension, a printer driver, or the Macintosh system software.
<code>gxPanelOtherEvt</code>	A different kind of event, such as an operating-system event, has occurred in the panel.
<code>gxPanelUserWillConfirmEvt</code>	The user has selected the confirm button, which means that it is time to parse panel interdependencies.

**Panel Responses**

A handler of a panel in a dialog box (including applications, printing extensions, printer drivers, and Macintosh system software) can return any value of type `OSErr` as the result of handling the panel. In addition, a panel handler can return an event of type `gxPanelResult`, as shown here. This data type is used with the `GXHandlePanelEvent` message, which is described on page 4-85.

```
enum {
    gxPanelNoResult          = 0,
    gxPanelCancelConfirmation = 1
};

typedef long gxPanelResult;
```

## Printing Messages

**Constant descriptions**`gxPanelNoResult`

The result field does not currently have any meaning.

`gxPanelCancelConfirmation`

This result is only valid if the panel event (as described in the previous section) was of type `gxPanelUserWillConfirmEvt`. After the user confirms the panel, if the panel handler discovers that the user entered an inappropriate value, the panel handler alerts the user to the problem and generates this response, which tells QuickDraw GX to not confirm the dialog box. This allows the user the opportunity to fix the problem.

**Panel Event Actions**

---

The panel event actions enumeration defines the constants used in the `evtAction` field of the panel information structure, which is described on page 4-35. Each value defines what action takes place after an event is processed.

```
enum {
    gxOtherAction          = 0,
    gxClosePanelAction     = 1,
    gxCancelDialogAction   = 2,
    gxConfirmDialogAction  = 3
};
```

**Constant descriptions**`gxOtherAction` The current item does not change after processing this event.`gxClosePanelAction`

The panel is closed after this event is processed.

`gxCancelDialogAction`

The dialog box is canceled after this event is processed.

`gxConfirmDialogAction`

The dialog box is confirmed after this event is processed.

**Parse Range Results**

---

The parse range results enumeration provides the constants that are used as the return result of the `GXParsePageRange` message, which is described on page 4-60.

```
enum {
    gxRangeNotParsed      = (gxParsePageRangeResult) 0,
    gxRangeParsed         = (gxParsePageRangeResult) 1,
    gxRangeBadFromValue   = (gxParsePageRangeResult) 2,
    gxRangeBadToValue     = (gxParsePageRangeResult) 3
};

typedef long gxParsePageRangeResult;
```



Printing Messages

Constant descriptions

<code>gxRangeNotParsed</code>	QuickDraw GX has not yet parsed a page range in the string.
<code>gxRangeParsed</code>	QuickDraw GX has successfully parsed a page range in the string.
<code>gxRangeBadFromValue</code>	QuickDraw GX has encountered an invalid value in the “from page” string during the parse.
<code>gxRangeBadToValue</code>	QuickDraw GX has encountered an invalid value in the “to page” string during the parse.

The Status Structure

The status structure, of data type `gxStatusRecord`, contains status information used with the status messages `GXJobStatus`, `GXWriteStatusToDTPWindow`, `GXInitializeStatusAlert`, `GXHandleAlertEvent`, and `GXHandleAlertStatus`. These messages are described in the section “Printing Messages Reference” beginning on page 4-9.

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

typedef struct gxStatusRecord gxStatusRecord;
```

Field descriptions

<code>statusType</code>	The type of status that this structure represents. This is one of the values shown in Table 4-1.
<code>statusId</code>	The ID of the status that this structure represents. If the value of this field is 0, there is no associated printing alert ('plrt') resource.
<code>statusAlertId</code>	The ID of the printing alert for this status.
<code>statusOwner</code>	The creator type of the owner of this status structure.
<code>statResId</code>	The resource ID for the status ('stat') resource used to process this status.
<code>statResIndex</code>	The index value for indexing into the status resource for this status.
<code>dialogResult</code>	The ID of the button string that was selected to dismiss the printing alert box associated with this status.

## Printing Messages

`bufferLen`      The number of bytes in the status buffer.

`statusBuffer`    This field is a buffer for the caller to store any additional information for use by the status-handling function.

**Note**

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

Table 4-1 shows the status types that you can specify in a status record.

**Table 4-1**      Status type IDs

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

## The Manual Feed Structure

The manual feed structure, of data type `gxManualFeedRecord`, is passed in the `statusBuffer` field of the status structure for a manual feed alert.

```
struct gxManualFeedRecord {
    Boolean   canAutoFeed;
    Str31     paperTypeName;
};

typedef struct gxManualFeedRecord gxManualFeedRecord;
```

## Printing Messages

**Field descriptions**

<code>canAutoFeed</code>	A Boolean value that is <code>true</code> when the driver can handle switching into automatic paper-feeding mode from manual-feed mode in the middle of a print job. This field is otherwise <code>false</code> .
<code>paperTypeName</code>	The string name of the paper type that is to be fed manually. This field can also be an array of paper-type names.

## The Display Structure

---

The display structure, of data type `gxDisplayRecord`, is used with the `GXWriteStatusToDTPWindow` message to send status information for display in the desktop printer window. The `GXWriteStatusToDTPWindow` message is described on page 4-163.

```
struct gxDisplayRecord {
    Boolean    useText;
    Handle     hPicture;
    Str255     theText;
};

typedef struct gxDisplayRecord gxDisplayRecord;
```

**Field descriptions**

<code>useText</code>	A Boolean value that specifies whether to use the text supplied in the <code>theText</code> field. If <code>true</code> , the text in the <code>theText</code> field is displayed; if <code>false</code> , the picture referred to by the <code>hPicture</code> field is displayed.
<code>hPicture</code>	A handle to the picture data to display. QuickDraw GX does not yet support this feature.
<code>theText</code>	The text string to display.

**Note**

The `hPicture` field is not yet used. At this time, you must set the value of `useText` to `true`. ♦

## Printing Messages

---

This section describes the messages that you can override to develop a printing extension or printer driver. This section presents these messages in categories that relate to the tasks for which you use each message.

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

## Printing Messages

A number of the printing messages can return communications result codes, which are listed in Table 4-2. The result codes section for each of these messages refers to this table.

**Table 4-2** Communications errors returned by many of the printing messages

Result code	Explanation
gxAioTimeout	The operation timed out
gxAioBdRqstState	Asynchronous communications are in an unexpected state for the operation
gxAioBadConn	The I/O connection reference number is not valid
gxAIOInvalidXfer	Bad values were discovered in the data transfer structure
gxAioNoRqstBlks	No request blocks are available to process the request
gxAioNoDataXfer	There is no pointer to a data transfer structure specified
gxAioTooManyAutos	Automatic status request is already active
gxAioNoAutoStat	The connection is not configured for automatic status request
gxAioBadRqstID	The I/O request identifier is not valid
gxAioCantKill	The communications protocol does not support I/O termination
gxAioAlreadyExists	The protocol-specific data was already specified
gxAioCantFind	The protocol-specific data was not found
gxAioDeviceDisconn	The printer was disconnected from the computer
gxAioNotImplemented	The specified function is not implemented
gxAioOpenPending	There is already a connection opened
gxAioNoProtocolData	There was not any protocol-specific data specified in the request
gxAioRqstKilled	The I/O request was terminated
gxBadBaudRate	The specified baud rate is not valid
gxBadParity	The specified parity value is not valid
gxBadStopBits	The specified stop-bits value is not valid
gxBadDataBits	The specified data-bits value is not valid
gxBadPrinterName	The specified printer name is not valid
gxAioBadMsgType	The message type specified in the data transfer structure is not valid
gxAioCantFindDevice	The target device could not be located
gxAioOutOfSeq	A non-atomic SCSI request was submitted in the wrong order

## Printing Messages

You must define each of your message overrides with the interface that is shown in each message description. Each override must match the formal declaration of the message that it is overriding: it must take the same parameters, in the same order, as shown in the example. You can provide your own name (whatever name you like) for each override that you define.

For clarity, Apple printing extensions and printer drivers apply the same prefix to each override they include. For example, the Apple Personal LaserWriter SC printer driver provides overrides named `SD_ShutDown`, `SD_DefaultPrinter`, `SD_DespoolPage`, and so on. The background picture printing extension includes overrides named `BWInitialize`, `BWShutdown`, and `BWDespoolPage`. You can choose to follow a similar convention for naming your message override functions. For example, you could name your override of the `GXCountPages` messages `MyCountPages`.

Most of the message descriptions also include a “Special Considerations” section, which describes whether you can send the message yourself and whether you can totally override the message. For messages that you can partially override, this section includes information about whether to forward the message prior to or after performing your own tasks.

## Storage Messages

---

Some messages are useful in all phases of printing and are needed by all message handlers. You can use these common messages to set up a “global” context for your handler. They allow you to save data between messages when needed.

## GXInitialize

---

QuickDraw GX sends the `GXInitialize` message at the start of a new printing job. You can override the `GXInitialize` message to perform any initialization your message handler needs to carry out its tasks. Your override of the `GXInitialize` message must match the following formal declaration:

```
OSErr MyInitialize (void);
```

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

### DESCRIPTION

QuickDraw GX sends the `GXInitialize` message at the start of a new printing job, just after the application has called the `GXNewJob` function.

## Printing Messages

You can override the `GXInitialize` message to perform any necessary initialization and to allocate storage for your extension or driver. `GXInitialize` is the first message a message handler receives after it is loaded into the message chain. If you need to allocate storage for global data in your override of `GXInitialize`, you need to call the `NewHandle` function and store the handle by calling the `SetMessageHandlerInstanceContext` function. Or you can call the `NewMessageGlobals` function to set up a globals environment to access so that you can access the global data. This stores the handle into the context for any other messages that you override. .

The default implementation of the `GXInitialize` message allocates memory needed by QuickDraw GX in its default implementation of the printing messages.

**SPECIAL CONSIDERATIONS**

You never send the `GXInitialize` message yourself.

You never forward the `GXInitialize` message.

**RESULT CODES**

<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**

You can find examples of overrides of the `GXInitialize` message in Listing 2-6 on page 2-16 in the chapter “Printing Extensions” and in Listing 3-4 on page 3-24 in the chapter “Printer Drivers.”

The `GXNewJob` function is described in *Inside Macintosh: QuickDraw GX Printing*.

The `NewHandle` function is described in *Inside Macintosh: Memory*. The `NewMessageGlobals` function is described in the chapter “Message Manager” in *Inside Macintosh: QuickDraw GX Environment and Utilities*.

The `SetMessageHandlerInstanceContext` function is described in *Inside Macintosh: Processes*.

**GXShutDown**

---

QuickDraw GX sends the `GXShutDown` message at the end of a print job. You can override the `GXShutDown` message to clean up or dispose of any private storage you allocated in your override of the `GXInitialize` message. Your override of the `Shutdown` message must match the following formal declaration:

```
OSErr MyShutDown (void);
```

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

#### DESCRIPTION

QuickDraw GX sends the `GXShutDown` message after a print job is complete. The printing context still exists when this message is sent.

You need to override this message to deallocate any storage you were using within the context of the print job and deallocate the handle you placed into the context as a result of the `GXInitialize` message. The `GXShutDown` message is always the last message a message handler receives before it is removed from the message chain.

The default implementation of the `GXShutDown` message deallocates memory that was allocated by the printing system.

#### SPECIAL CONSIDERATIONS

You need to override the `GXShutDown` message and deallocate storage if you have also overridden the `GXInitialize` message.

You never send the `GXShutDown` message yourself.

You never forward the `GXShutDown` message to other message handlers.

#### RESULT CODES

<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.
<code>memFullErr</code>	There is not enough memory to retrieve the tagged data.
<code>resNotFound</code>	The specified data could not be found.

#### SEE ALSO

You can find examples of overrides of the `GXShutDown` message in Listing 2-18 on page 2-31 in the chapter “Printing Extensions” and in Listing 3-18 on page 3-48 in the chapter “Printer Drivers.”

The `GXInitialize` message is described in the previous section.

## GXFetchTaggedData

QuickDraw GX, an application, a printing extension, or a printer driver can send the `GXFetchTaggedData` message to retrieve or modify data from a resource. You can override the `GXFetchTaggedData` message to modify data from your resources at run time. Use `GXFetchTaggedData` to retrieve and modify resource data that you access by

## Printing Messages

resource type and ID. Your override of the `GXFetchTaggedData` message must match the following formal declaration:

```
OSErr MyFetchTaggedData (ResType aResType, short id,
                        Handle *aHandle, Signature owner);
```

<code>aResType</code>	On input, the resource type in which you are interested.
<code>id</code>	The resource ID.
<code>aHandle</code>	On return, a pointer to a handle that contains the resource data.
<code>owner</code>	The owner of the requested data.

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

## DESCRIPTION

QuickDraw GX or any message handler can send this message to obtain resource information from a printer driver. You can send this message to yourself to obtain data from a particular resource.

You specify the resource type and ID of the resource to retrieve in the `aResType` and `id` parameters, respectively. The resource handle is returned with a handle value in the `aHandle` parameter. The `owner` parameter defines the owner type of the resource.

The default implementation of this message accesses the resource that belongs to the driver by calling the `GetResource` function and then calling the `DetachResource` function. Override this message if you wish to change resource data that is being read from a printing extension, a printer driver, or the desktop printer.

If you are not the owner of the data (that is, if the owner ID is not your creator type), you need to forward this message. If the owner ID is `'drvvr'`, the message is intended for the currently active driver. If you are the owner, you need to create a handle for the tagged data and fill it out. If you use the `GetResource` function to get the handle, be sure to call the `DetachResource` function to change the handle into a non-Resource Manager handle.

## RESULT CODES

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

## SEE ALSO

The `GetResource` and `DetachResource` functions are described in the chapter “Resource Manager” in *Inside Macintosh: More Macintosh Toolbox*.



## Print Object Messages

---

Print object messages allow you to manipulate the various print objects that are created and referenced by an application during all phases of the printing process. These objects are the job object, the format object, and the paper-type object. You can choose to override print object messages to add additional data to these objects. You can also change the default information to something that has more meaning for you.

## GXDefaultJob

---

QuickDraw GX sends the `GXDefaultJob` message when an application creates a new job. You can override the `GXDefaultJob` message to add to or modify the contents of the job object. Your override of the `GXDefaultJob` message must match the following formal declaration:

```
OSErr MyDefaultJob (void);
```

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

### DESCRIPTION

QuickDraw GX sends the `GXDefaultJob` message when an application calls the `GXNewJob` function to create a new job object.

You override this message if you need to add additional data to the job object when it is created. After you forward this message down the message chain, you can add your own information to the job object or change information that was placed there by the default implementation.

The default implementation of this message provides the default information for this job object.

### SPECIAL CONSIDERATIONS

You can find an example of an override of the `GXDefaultJob` message in Listing 3-6 on page 3-28 in the chapter “Printer Drivers.”

You never send the `GXDefaultJob` message yourself.

You must forward the `GXDefaultJob` message to other message handlers. Always forward it before you add or change job information.

## Printing Messages

## RESULT CODES

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

## SEE ALSO

The `GXNewJob` function is described in *Inside Macintosh: QuickDraw GX Printing*.

## GXDefaultFormat

---

QuickDraw GX sends the `GXDefaultFormat` message when an application creates a new format object. You can override the `GXDefaultFormat` message to modify or add to the contents of a newly created format object. Your override of the `GXDefaultFormat` message must match the following formal declaration:

```
OSErr MyDefaultFormat (gxFormat aFormat);
```

`aFormat`      The format object.

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

## DESCRIPTION

QuickDraw GX sends the `GXDefaultFormat` message when an application calls the `GXNewFormat` function to create a new format.

You override this message if you need to add additional data to the format object when it is created. After you forward this message down the message chain, you can add your own information to the format object or change information that was placed there by the default implementation. The default implementation of this message provides the default information for this format object.

## SPECIAL CONSIDERATIONS

You never send the `GXDefaultFormat` message yourself.

You must forward the `GXDefaultFormat` message to other message handlers. Always forward it before you add or change format information.

**RESULT CODES**

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

**SEE ALSO**

The `GXNewFormat` function is described in *Inside Macintosh: QuickDraw GX Printing*.

**GXDefaultPaperType**

---

QuickDraw GX sends the `GXDefaultPaperType` message when an application creates a new paper-type object. You can override the `GXDefaultPaperType` message to make changes in the paper type used by a printing job. Your override of the `GXDefaultPaperType` message must match the following formal declaration:

```
OSErr MyDefaultPaperType (gxPaperType aPaperType);
```

`aPaperType` A paper-type object to fill in with the default value.

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

**DESCRIPTION**

QuickDraw GX sends the `GXDefaultPaperType` message when an application calls the `GXNewPaperType` function to create a new paper type.

You override this message if you need to add additional data to the paper-type object at the time that it is created or if you wish to change the default paper type to another type. After you forward this message down the message chain, you can add your own information to the paper-type object or change information that was placed there by the default implementation. The default implementation of this message provides the default information for this paper type, including size, margins, and PostScript data.

**SPECIAL CONSIDERATIONS**

You never send the `GXDefaultPaperType` message yourself.

You must forward the `GXDefaultPaperType` message to other message handlers. Always forward it before you add or change paper-type information.

## Printing Messages

## 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.
<code>gxPaperTypeNotFound</code>	The specified paper type could not be found.
<code>gxNoSuchPTGroup</code>	The specified paper type could not be found.

## SEE ALSO

The `GXNewPaperType` function is described in *Inside Macintosh: QuickDraw GX Printing*.

**GXDefaultPrinter**

---

QuickDraw GX sends the `GXDefaultPrinter` message when an application creates a new printer object. You can override the `GXDefaultPrinter` message to modify the default printer object as it is being created. Your override of the `GXDefaultPrinter` message must match the following formal declaration:

```
OSErr MyDefaultPrinter (gxPrinter aPrinter);
```

`aPrinter`     The printer object.

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

## DESCRIPTION

QuickDraw GX sends the `GXDefaultPrinter` message when an application calls the `GXNewJob` function to create a new job object.

You override this message if you need to modify the default printer object at the time that it is created. For example, you may want your printer driver to supply an application with the list of the valid color spaces and printing devices that it supports. After you forward this message down the message chain, you can add your own information to the printer object or change information that was placed there by the default implementation. The default implementation of this message provides the default information for the printer object.

## SPECIAL CONSIDERATIONS

You never send the `GXDefaultPrinter` message yourself.

You must forward the `GXDefaultPrinter` message to other message handlers. Always forward it before you add or change printer information.

**RESULT CODES**

<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**

You can find an example of an override of the `GXDefaultPrinter` message in Listing 3-5 on page 3-25 in the chapter “Printer Drivers.”

The `GXNewJob` function is described in *Inside Macintosh: QuickDraw GX Printing*.

**GXDefaultDesktopPrinter**

---

QuickDraw GX sends the `GXDefaultDesktopPrinter` message when the user creates a new desktop printer with the Chooser. You can override the `GXDefaultDesktopPrinter` message to modify the configuration file of the desktop printer. Your override of the `GXDefaultDesktopPrinter` message must match the following formal declaration:

```
OSErr MyDefaultDesktopPrinter (Str31 dtpName);
```

`dtpName`      The name of the desktop printer.

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

**DESCRIPTION**

QuickDraw GX sends the `GXDefaultDesktopPrinter` message when a user creates a new desktop printer.

You override this message if you need to add additional data to the printer's configuration file. For example, you may want your driver to supply an application with the list of the valid color spaces and printing devices that it supports or you might want to fill in the default configuration for the printer. After you forward this message down the message chain, you can add your own information to the printer object or change information that was placed there by the default implementation. The default implementation of this message fills in the default information for the printer object.

**SPECIAL CONSIDERATIONS**

You never send the `GXDefaultDesktopPrinter` message yourself.

You must forward the `GXDefaultDesktopPrinter` message to other message handlers. Always forward it before you add or change configuration information.

## Printing Messages

## 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

Creating desktop printers is described in *Inside Macintosh: QuickDraw GX Printing*.

Application Messages

---

QuickDraw GX sends application messages in response to an application calling QuickDraw GX functions to accomplish printing-related tasks. You rarely need to override these messages because modifying the data prior to spooling is more easily done by overriding spooling messages, which are described in the section “Spooling Messages” beginning on page 4-67.

GXStartJob

---

QuickDraw GX sends the `GXStartJob` message when the spooling of a document is initiated. You can override the `GXStartJob` message to set initial values at the start of printing a document. Your override of the `GXStartJob` message must match the following formal declaration:

```
OSErr MyStartJob (StringPtr docName, long pageCount);
```

`docName`      The document name.

`pageCount`    The number of pages in the document.

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

## DESCRIPTION

QuickDraw GX sends the `GXStartJob` message to initiate spooling when an application calls the `GXStartJob` function to start printing.

You need to override this message if you want to initialize information when an application begins printing a document. You can also override this message to determine when a document is being spooled.

The default implementation of `GXStartJob` begins the process of saving the document into a spool file. It sends spooling messages to accomplish this.

**SPECIAL CONSIDERATIONS**

You never send this message yourself.

You must forward the `GXStartJob` message to other message handlers so that they can override it. If your override fails, you need to call the `GXCleanupStartJob` function to notify other handlers of the failure. If another handler returns an error, you must undo anything that you've done and return the same error.

**RESULT CODES**

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

**SEE ALSO**

The `GXCleanupStartJob` function is described on page 5-36 in the chapter “Printing Functions for Message Overrides.”

**GXCleanupStartJob**

---

QuickDraw GX sends the `GXCleanupStartJob` message if a message handler fails during the processing of a `GXStartJob` message. You can override the `GXCleanupStartJob` message to deallocate any storage you allocated in your override of the `GXStartJob` message. Your override code for `GXCleanupStartJob` has no parameters and returns no value. For example, you could declare your override for the `GXCleanupStartJob` message as follows:

```
void MyCleanupStartJob (void);
```

**DESCRIPTION**

QuickDraw GX sends the `GXCleanupStartJob` message after a message handler calls the `GXCleanupStartJob` function.

You need to override this message to deallocate any storage that you allocated in your override of the `GXStartJob` message.

The default implementation of this message disposes of memory that was allocated for the printing job.

**SPECIAL CONSIDERATIONS**

You never send the `GXCleanupStartJob` message yourself; however, you can call the `GXCleanupStartJob` function, which then sends this message.

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

**SEE ALSO**

The `GXStartJob` message is described in the previous section.

The `GXCleanupStartJob` function is described on page 5-36 in the chapter “Printing Functions for Message Overrides.”

**GXFinishJob**

---

QuickDraw GX sends the `GXFinishJob` message when the spooling of a document is finished. You can override the `GXFinishJob` message to perform operations required by your printing extension or printer driver at the completion of spooling a document. Your override of the `GXFinishJob` message must match the following formal declaration:

```
OSErr MyFinishJob (void);
```

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

**DESCRIPTION**

QuickDraw GX sends the `GXFinishJob` message when an application calls the `GXFinishJob` function to indicate that spooling of a document is complete.

You need to override this message if you want to know when a document has finished spooling or to finalize information when spooling of a document is done.

The default implementation of `GXFinishJob` completes the spooling process by sending spooling messages. It also updates the job object data to contain the correct number of pages that are to be printed. It sends spooling messages to accomplish this.

**SPECIAL CONSIDERATIONS**

You never send the `GXFinishJob` message yourself.

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

**RESULT CODES**

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

**SEE ALSO**

The `GXFinishJob` function is described in *Inside Macintosh: QuickDraw GX Printing*.



## GXJobIdle

---

QuickDraw GX sends the `GXJobIdle` message when a printer driver, a printing extension, or an application calls the `GXJobIdle` function to release idle time. You can override the `GXJobIdle` message to obtain time from the idle procedure. Your override of the `GXJobIdle` message must match the following formal declaration:

```
OSErr MyJobIdle (void)
```

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

### DESCRIPTION

You can override this message to perform tasks during idle time. The default implementation of this message gives up time to other processes.

### SPECIAL CONSIDERATIONS

You never send the `GXJobIdle` message yourself; however, you can call the `GXJobIdle` function, which then sends this message.

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

### RESULT CODES

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

### SEE ALSO

The `GXJobIdle` function is described on page 5-33 in the chapter “Printer Functions for Message Overrides.”

## GXStartPage

---

QuickDraw GX sends the `GXStartPage` message to start a new page in the spool file. You can override the `GXStartPage` message to perform any initialization your printing extension or printer driver requires before printing each page. Your override of the `GXStartPage` message must match the following formal declaration:

```
OSErr MyStartPage (gxFormat aFormat, long numViewPorts,
                  gxViewport *viewPortList);
```

`aFormat` The format object for the page.

## Printing Messages

`numViewPorts`

The number of view ports pointed to by the `viewPortList` parameter.

`viewPortList`

A pointer to a list of view ports to use to capture shapes.

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

## DESCRIPTION

QuickDraw GX sends this message when an application calls the `GXStartPage` function to start a new page. The application calls the `GXStartPage` and `GXFinishPage` functions once for each page, which causes QuickDraw GX to send the `GXStartPage` and `GXFinishPage` messages. The `GXStartPage` message begins a new page in the spool file and prepares to capture all data drawn to the view ports in the `viewPortList` parameter so that this data can be redirected to the new page.

You need to override this message if you want to initialize information when an application begins printing each page. You can also override this message to determine when a new page is being spooled.

The default implementation of `GXStartPage` installs view port filters on the requested view ports to begin capturing graphics objects.

## SPECIAL CONSIDERATIONS

You never send the `GXStartPage` message yourself.

You must forward the `GXStartPage` message to other message handlers so that they can override it. If your override fails, you need to call the `GXCleanupStartPage` function to notify other handlers of the failure. If another handler returns an error, you must undo anything that you've done and return the same error.

## RESULT CODES

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

## SEE ALSO

The `GXStartPage` and `GxFinishPage` functions are described in *Inside Macintosh: QuickDraw GX Printing*.

The `GXCleanupStartPage` function is described on page 5-37 in the chapter “Printing Functions for Message Overrides.”

## GXCleanupStartPage

---

QuickDraw GX sends the `GXCleanupStartPage` message if a message handler fails during the processing of a `GXStartPage` message. You can override the `GXCleanupStartPage` message to deallocate any storage allocated during processing of the `GXStartPage` message. Your override of the `GXCleanupStartPage` message must match the following formal declaration:

```
void MyCleanupStartPage (void);
```

### DESCRIPTION

QuickDraw GX sends the `GXCleanupStartPage` message after it receives a `GXCleanupStartPage` function call from a message handler that failed during a `GXStartPage` override and after forwarding the `GXStartPage` message.

You need to override the `GXCleanupStartPage` message if you perform an operation that must be undone within the `GXStartPage` message override. For example, your attempt to allocate memory or initialize a device after forwarding the `GXStartPage` message may fail. When you receive this message, you typically deallocate any storage you were using because of `GXStartPage`.

The default implementation of `GXCleanupStartPage` disposes of the memory that was allocated during the default implementation of the `GXStartPage` message.

### SPECIAL CONSIDERATIONS

You never send the `GXCleanupStartPage` message yourself; however, you can call the `GXCleanupStartPage` function, which then sends this message.

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

### SEE ALSO

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

The `GXCleanupStartPage` function is described on page 5-37 in the chapter “Printing Functions for Message Overrides.”

## GXFinishPage

---

QuickDraw GX sends the `GXFinishPage` message when spooling of a page is finished. You can override the `GXFinishPage` message to perform any action required at the end of each page. Your override of the `GXFinishPage` message must match the following formal declaration:

```
OSErr MyFinishPage (void);
```

## Printing Messages

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

**DESCRIPTION**

QuickDraw GX sends this message when an application calls the `GXFinishPage` function to complete a page.

You need to override this message if you want to perform an action at the end of each page. QuickDraw GX sends the `GXFinishPage` message once for each `GXStartPage` message that is sent. This message indicates that the data has been drawn and that it can be written to the spool file.

The default implementation of `GXFinishPage` completes the associated page and spools the data for the page by sending the `GXSpoolPage` message. It also releases the view ports that were captured when the `GXStartPage` message was sent.

**SPECIAL CONSIDERATIONS**

You never send the `GXFinishPage` message yourself.

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

**RESULT CODES**

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

**SEE ALSO**

The `GXFinishPage` function is described in *Inside Macintosh: QuickDraw GX Printing*.

**GXPrintPage**

---

QuickDraw GX sends the `GXPrintPage` message when the data for a page is ready to be spooled. You can override the `GXPrintPage` message to modify data sent by an application for printing. Your override of the `GXPrintPage` message must match the following formal declaration:

```
void MyPrintPage (gxFormat aFormat, gxShape aPage);
```

`aFormat`      The format object for the page.

`aPage`        The data that belongs on the page in the form of a picture shape.

**DESCRIPTION**

QuickDraw GX sends this message when an application call the `GXPrintPage` function to print a page.

You rarely need to override this message. If you wish to modify the data that the application sends, it is preferable to override the `GXSpoolPage` message. This works for applications that use `GXPrintPage` for printing and also for those that use the `GXStartPage`/`GXDrawShape`/`GXFinishPage` sequence for printing.

**SPECIAL CONSIDERATIONS**

You never send the `GXPrintPage` message yourself.

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

**SEE ALSO**

The `GXPrintPage` function is described in *Inside Macintosh: QuickDraw GX Printing*.

## **GXJobFormatModeQuery**

---

QuickDraw GX sends the `GXJobFormatModeQuery` message when the application requests or sets format mode information. You can override the `GXJobFormatModeQuery` message to provide support for direct modes. Your override of the `GXJobFormatModeQuery` message must match the following formal declaration:

```
void MyJobFormatModeQuery (gxQueryType aQueryType,
                           void *srcData, void *dstData);
```

`aQueryType`    The type of query being performed.

`srcData`        A pointer to the input data for the query.

`dstData`        A pointer to the output (result) data for the query.

**DESCRIPTION**

QuickDraw GX sends the `GXJobFormatModeQuery` message when an application calls the `GXJobFormatModeQuery` function to get or set additional format-mode information.

The type of query made by the application is specified in the `aQueryType` parameter, the possible values of which are listed and described in the “Advanced Printing Features” chapter in *Inside Macintosh: QuickDraw GX Printing*. Use of the `srcData` and `dstData` parameters is described in the same chapter.

You only need to override this message if you support job format modes. Based on the query from the application, you provide the requested information.

## Printing Messages

The default implementation of the `GXJobFormatModeQuery` message does nothing.

**SPECIAL CONSIDERATIONS**

You never send the `GXJobFormatModeQuery` message yourself.

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

**SEE ALSO**

You can find an example of an override of the `GXJobFormatModeQuery` message in Listing 3-10 on page 3-34 in the chapter “Printer Drivers.”

The `GXJobFormatModeQuery` function, including the types of queries that you can make and the use of the `srcData` and `dstData` parameters, is described in *Inside Macintosh: QuickDraw GX Printing*.

**GXParsePageRange**

---

QuickDraw GX sends the `GXParsePageRange` message when a user selects a range of pages for printing. You can override the `GXParsePageRange` message to validate a page range. Your override of the `GXParsePageRange` message must match the following formal declaration:

```
OSErr MyParsePageRange (StringPtr fromString, StringPtr toString,
                        gxParsePageRangeResult *result);
```

`fromString` A pointer to a string representation of the from-page.

`toString` A pointer to a string representation of the to-page.

`result` On return, a value that specifies the result code for the range parsing. The constants for this value are given in the section “Parse Range Results” on page 4-38.

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

**DESCRIPTION**

QuickDraw GX sends the `GXParsePageRange` message to validate that a page range entered by the user is appropriate for the print job.

**SPECIAL CONSIDERATIONS**

You rarely send the `GXParsePageRange` message yourself.

You must always forward the `GXParsePageRange` message to other message handlers.

**RESULT CODES**

<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.

**Paper-Handling Messages**

---

QuickDraw GX provides the `GXDoesPaperFit` message to determine if a specific paper type can be used on a printer. QuickDraw GX also provides several paper-handling functions, each of which is described in the chapter “Printing Functions for Message Overrides” in this book.

**GXDoesPaperFit**

---

QuickDraw GX sends the `GXDoesPaperFit` message to determine if a specific paper type is acceptable on the printing device that your driver supports. You can override the `GXDoesPaperFit` message to set or modify the returned value. Your override of the `GXDoesPaperFit` message must match the following formal declaration:

```
OSErr MyDoesPaperFit (gxTrayIndex whichTray, gxPaperType paper,
                     Boolean *fits);
```

`whichTray`    The tray in which the paper is to be loaded.

`paper`        The paper type.

`fits`         On return, a Boolean value that is `true` if the paper can be used on the printer and `false` if not.

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

**DESCRIPTION**

QuickDraw GX sends the `GXDoesPaperFit` message to limit the number of paper-type names that appear in the Paper pop-up menu in the Page Setup dialog box. You can override the `GXDoesPaperFit` message to notify QuickDraw GX that a certain paper type does not work in the specified paper tray on your device.

**RESULT CODES**

<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.

## Color Profile Messages

---

QuickDraw GX sends color profile messages to allow printing extensions and printer drivers to work with the color profiles that are used for color matching. Color matching and the ColorSync Manager are described in *Inside Macintosh: Advanced Color Imaging*.

## GXFindPrinterProfile

---

QuickDraw GX sends the `GXFindPrinterProfile` message to allow you to modify color-matching information for your printer driver. Your override of the `GXFindPrinterProfile` message must match the following formal declaration:

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

`thePrinter`

The printer object.

`searchData`

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

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

`index`

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

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

`returnedProfile`

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

`numProfiles`

On return, the number of profiles that were found.

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



**DESCRIPTION**

When an application calls the `GXFindPrinterProfile` function to search for a color profile that matches certain specifications, QuickDraw GX sends the `GXFindPrinterProfile` message to allow your printing extension or printer driver to modify the search criteria or results.

The default implementation of the `GXFindPrinterProfile` message checks the profile that is associated with the view device of the current printer. If no profile match is made, it then checks the profile that is associated with the color profile ('prof') resource that is stored with the desktop printer. If no profile is matched, it then checks the profile that is associated with the color profile resource that is stored with the printer driver. If no match is made, then `nil` is returned in the `returnedProfile` parameter. You can override this message to change the profile matching data prior to forwarding the message or to modify the match results after forwarding the message.

**RESULT CODES**

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

**SEE ALSO**

The `GXFindPrinterProfile` function is described in *Inside Macintosh: QuickDraw GX Printing*.

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

**GXFindFormatProfile**

QuickDraw GX sends the `GXFindFormatProfile` message to allow you to modify color matching information for a specific format object. This message is similar to the `GXFindPrinterProfile` message (described above), except that it finds a color profile that is associated with a format object rather than a printer object. Your override of the `GXFindFormatProfile` message must match the following formal declaration:

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

`theFormat` The format object.

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

## Printing Messages

	If this value is <code>nil</code> , the value of the <code>index</code> parameter defines which profile is returned.
<code>index</code>	<p>The index of the profile to return. If the value is 0, then the current profile is returned in the <code>returnedProfile</code> parameter.</p> <p>If the value of this parameter is not 0, then the behavior this function depends on the value of the <code>searchData</code> parameter. If <code>index</code> is not 0 and <code>searchData</code> is <code>nil</code>, the indexed profile is returned in the <code>returnedProfile</code> parameter. If <code>index</code> is not 0 and <code>searchData</code> is not <code>nil</code>, then the printer profiles are searched.</p>
<code>returnedProfile</code>	On return, a color profile matching the criteria specified by the <code>searchData</code> and <code>index</code> parameters. If no color profiles are found, this value is <code>nil</code> upon return.
<code>numProfiles</code>	On return, the number of profiles that were found.
<b>function result</b>	An error code. The value <code>noErr</code> indicates that the operation was successful.

## DESCRIPTION

When an application calls the `GXFindFormatProfile` function to search for a color profile that matches certain specifications, QuickDraw GX sends the `GXFindFormatProfile` message to allow your printing extension or printer driver to modify the search criteria or results.

The default implementation of the `GXFindFormatProfile` message checks the profile that is associated with the specified format object. If no profile match is made, it then checks the profile that is associated with the color profile ('prof') resource that is stored with the desktop printer. If no profile is match, it then checks the profile that is associated with the color profile resource that is stored with the printer driver. If no match is made, then `nil` is returned in the `returnedProfile` parameter. You can override this message to change the profile matching data prior to forwarding the message or to modify the match results after forwarding the message.

## RESULT CODES

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

## SEE ALSO

The `GXFindFormatProfile` function is described in *Inside Macintosh: QuickDraw GX Printing*.

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

## GXSetPrinterProfile

QuickDraw GX sends the `GXSetPrinterProfile` message to change the current color profile for a printer. Your override of the `GXSetPrinterProfile` message must match the following formal declaration:

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

`thePrinter` The printer object.

`oldProfile` The profile that has been associated with the printer object.

`newProfile` The profile to add to the list of profiles for a printer object.

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

### DESCRIPTION

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

The values of the `oldProfile` and `newProfile` parameters define what happens in response to this message, as shown in Table 4-3.

**Table 4-3** The actions of the `GXSetPrinterProfile` message

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

The default implementation of this message modifies the list of profiles that are associated with the printer object, as shown in Table 4-3.

## Printing Messages

## RESULT CODES

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

## SEE ALSO

The `GXSetPrinterProfile` function is described in *Inside Macintosh: QuickDraw GX Printing*.

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

## GXSetFormatProfile

---

QuickDraw GX sends the `GXSetFormatProfile` message to change the current color profile for a format object. Your override of the `GXSetFormatProfile` message must match the following formal declaration:

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

`theFormat` The format object.

`oldProfile` The profile that has been associated with the format object.

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

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

## DESCRIPTION

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

Printing Messages

The values of the `oldProfile` and `newProfile` parameters define what happens in response to this message, as shown in Table 4-3.

**Table 4-4** The actions of the `GXSetFormatProfile` message

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

The default implementation of this message modifies the list of profiles that are associated with the format object, as shown in Table 4-3.

RESULT CODES

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

SEE ALSO

The `GXSetFormatProfile` function is described in *Inside Macintosh: QuickDraw GX Printing*.

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

Spooling Messages

When an application sends shapes to be printed, QuickDraw GX uses spooling messages to send these shapes to spool files. It sends these messages when the application is calling QuickDraw GX functions to accomplish printing-related tasks. You can override spooling messages to change the data as it is being written to disk.

Some printing extensions and printer drivers need to perform their own spooling or create a custom format for the spool file. If this is the case for your extension or driver, you need to override all of the spooling and despooling messages.

When you override a spooling message, you need to choose whether to forward the message to the other handlers in the message chain. If you are performing your own spooling and overriding all of the spooling and despooling messages, you don't need to

## Printing Messages

forward the messages. However, if you are not handling all of the spooling yourself, you do need to forward each message so that the default implementation can perform its task.

## GXCreateSpoolFile

---

QuickDraw GX sends the `GXCreateSpoolFile` message at the start of spooling a document. You can override the `GXCreateSpoolFile` message to perform the setup for spool files. Your override of the `GXCreateSpoolFile` message must match the following formal declaration:

```
OSErr MyCreateSpoolFile (FSSpecPtr aFSSpecPtr,
                        long createOptions, gxSpoolFile *aSpoolFile);
```

**aFSSpecPtr** A pointer to a file specification that indicates where the spool file is created.

**createOptions** Options for creating the spool file, as shown in Table 4-5.

**aSpoolFile** A pointer to a file specification, which on return is a value that references the newly created spool file.

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

### DESCRIPTION

QuickDraw GX sends the `GXCreateSpoolFile` message at the start of spooling, when an application calls the `GXStartJob` function to create a new spool file. Several options for creating the file may be specified by QuickDraw GX, as shown in Table 4-5.

**Table 4-5** Options for the `GXCreateSpoolFile` message

Constant	Value	Explanation
<code>gxNoCreateOptions</code>	<code>0x00000000</code>	No options for this file: just create it.
<code>gxInhibitAlias</code>	<code>0x00000001</code>	Do not create an alias for this file in the Print Monitor Documents (PMD) folder.
<code>gxInhibitUniqueName</code>	<code>0x00000002</code>	Do not append anything to the filename to make it unique. If you specify this option and the file already exists, the file is replaced.
<code>gxResolveBitmapAlias</code>	<code>0x00000004</code>	Include bitmap data instead of the alias for the bitmap data.

## Printing Messages

If you override the `GXCreateSpoolFile` message, you can perform any setup you wish. You can override this message to change the location of the spool file by changing the contents of the `aFSSpecPtr` parameter. You can also override this message and all subsequent spooling messages if you want to do all your own spooling. The code in your override of this message is executed once per spooled job.

**SPECIAL CONSIDERATIONS**

You rarely send the `GXCreateSpoolFile` message yourself.

If you are providing your own spooling, you need to totally override the `GXCreateSpoolFile` message and all of the other spooling and despooling messages.

If you are not providing your own spooling (which is almost always the case), you must forward the `GXCreateSpoolFile` message because QuickDraw GX's default implementation creates a new spool file that is used by the other spooling messages.

**RESULT CODES**

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

**SEE ALSO**

The `GXStartJob` function is described in *Inside Macintosh: QuickDraw GX Printing*.

**GXSpoolPage**

QuickDraw GX sends the `GXSpoolPage` message when a page is ready to be sent to the spool file. You can override the `GXSpoolPage` message to change or add to a page being sent to a spool file. Your override of the `GXSpoolPage` message must match the following formal declaration:

```
OSErr MySpoolPage (gxSpoolFile aSpoolFile, gxFormat aFormat,
                  gxShape aShape);
```

`aSpoolFile` The file to which the page is being written.

`aFormat` The format that goes with the page.

`aShape` The data that belongs on the page in the form of a picture shape.

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

## Printing Messages

## DESCRIPTION

QuickDraw GX sends the `GXSpoolPage` message when an application calls either the `GXFinishPage` function or the `GXPrintPage` function. This message takes a picture shape in the `aShape` parameter that represents a page and writes it to the spool file referenced by the `aSpoolFile` parameter.

You override this message when you need to perform any per-page operations such as adding data (for example, a background picture or confidential stamp) to each page. You need to change the page before forwarding this message because the page is written to the file before control returns from the forwarded message.

The default implementation of `GXSpoolPage` writes the data into the spool file in a standard format.

## SPECIAL CONSIDERATIONS

You rarely send the `GXSpoolPage` message yourself.

If you are providing your own spooling, you need to totally override `GXSpoolPage` and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward this message to allow the default implementation to write the data into the spool file.

## RESULT CODES

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

## SEE ALSO

The `GXFinishPage` and `GXPrintPage` functions are described in *Inside Macintosh: QuickDraw GX Printing*.

## GXSpoolData

---

QuickDraw GX sends the `GXSpoolData` message whenever a stream of data is about to be written to the spool file. You can override the `GXSpoolData` message to modify data going into the spool file or to spool data in your own way. Your override of the `GXSpoolData` message must match the following formal declaration:

```
OSErr MySpoolData (gxSpoolFile aSpoolFile, Ptr data,
                  long *length);
```

`aSpoolFile`

The spool file to which the data is written.



## Printing Messages

<code>data</code>	A pointer to the buffer containing the data.
<code>length</code>	On entry, the length of the buffer in bytes. On return, the actual number of bytes that were spooled.

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

**DESCRIPTION**

QuickDraw GX sends the `GXSpoolData` message during spooling when a stream of data is about to be written to the spool file.

One reason to override this message is to encrypt the data in the spool file. If you do change the data in the spool file with an override of the `GXSpoolData` message, you must do so prior to forwarding the message.

The default implementation of the `GXSpoolData` message stores the data into the spool file. In most cases, you forward the `GXSpoolData` message after making your data changes and let the default implementation write the data bytes.

**SPECIAL CONSIDERATIONS**

You rarely send the `GXSpoolData` message yourself.

If you are providing your own spooling, you need to totally override the `GXSpoolData` message and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward the `GXSpoolData` message to allow the default implementation to write the data into the spool file.

**RESULT CODES**

<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.

**GXSpoolResource**

QuickDraw GX sends the `GXSpoolResource` message when a resource is about to be added to the spool file. You can override the `GXSpoolResource` message to add your own resource to the spool file. Your override of the `GXSpoolResource` message must match the following formal declaration:

```
OSErr MySpoolResource (gxSpoolFile aSpoolFile, Handle aResource,
                      ResType aType, short id);
```

`aSpoolFile` The spool file to which you are adding resources.

## Printing Messages

`aResource` A handle to the resource that you want added to the spool file.  
`aType` The resource type of the resource that you are adding.  
`id` The resource ID of the resource that you are adding.

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

## DESCRIPTION

QuickDraw GX sends the `GXSpoolResource` message after a message handler has issued a `GXSpoolResource` message to write a resource to the spool file.

You can override this message if you need to change the data in a resource that is being added to the spool file. For example, if you are encrypting the data in a spool file, you might want to override this message and encrypt the resource data before it is written to the file.

The default implementation of `GXSpoolResource` expects the `aResource` handle to be a normal memory handle, not a resource. After this message executes, the handle has become a resource handle that the caller cannot use. This behavior is the same as for the `AddResource` function, which is described in the chapter “Resource Manager” in *Inside Macintosh: More Macintosh Toolbox*.

## SPECIAL CONSIDERATIONS

You can send the `GXSpoolResource` message yourself if you have a resource that you want to store in the spool file.

If you are providing your own spooling, you need to totally override the `GXSpoolResource` message and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward the `GXSpoolResource` message to allow the default implementation to write the data into the spool file.

## RESULT CODES

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

## GXCompleteSpoolFile

---

QuickDraw GX sends the `GXCompleteSpoolFile` message when spooling of a document has been completed and the spool file is about to be closed. You can override the `GXCompleteSpoolFile` message to conclude operations on a spool file. Your override of the `GXCompleteSpoolFile` message must match the following formal declaration:

```
OSErr MyCompleteSpoolFile (gxSpoolFile aSpoolFile);
```

`aSpoolFile` The spool file.

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

### DESCRIPTION

QuickDraw GX, in its default implementation of the `GXFinishJob` message, sends the `GXCompleteSpoolFile` message when spooling is complete and the spool file is about to be closed.

You can override this message to determine when a spool file is closing or to conclude your operations on the spool file. You can also override this message if you have code you wish to execute once per spooled job.

The default implementation of the `GXCompleteSpoolFile` message finishes the spooling process and closes the file.

### SPECIAL CONSIDERATIONS

You rarely send the `GXCompleteSpoolFile` message yourself.

If you are providing your own spooling, you need to totally override the `GXCompleteSpoolFile` message and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward the `GXCompleteSpoolFile` message to allow the default implementation to finish spooling data and close the spool file.

### RESULT CODES

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

### SEE ALSO

The `GXFinishJob` message is described on page 4-54.

## Despooling Messages

---

Despooling messages are sent during the imaging phase of printing to read the data from the disk and prepare it for rendering into a format that is acceptable for the printing device. You can override these messages to modify or change data just prior to it being rendered.

## GXCountPages

---

QuickDraw GX sends the `GXCountPages` message to determine how many pages are in the file that is about to be despoiled. You can override the `GXCountPages` message to change the logical number of pages in the spool file. Your override of the `GXCountPages` message must match the following formal declaration:

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

`aSpoolFile`    The spool file.

`numPages`     On return, the number of pages in the spool file.

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

### DESCRIPTION

QuickDraw GX, in its default implementation of the `GXImageDocument` message, sends the `GXCountPages` message to determine the number of pages in the spool file prior to despooling.

You override this message if you need to change the logical number of pages in the spool file (for example, if you are producing thumbnail sketches of the document page with several to a printed page).

The default implementation of this message counts and returns the number of pages in the spool file. If you need to alter the page count returned by this message, you need to first forward the message and then add your own code, as you would in the case of the thumbnail example.

### SPECIAL CONSIDERATIONS

You rarely send the `GXCountPages` message yourself.

You must forward the `GXCountPages` message because QuickDraw GX depends on receiving it to know how many pages to despool. You need to first forward this message and then change the value, if necessary.

**RESULT CODES**

<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.
<code>gxIncompletePrintFileErr</code>	The spool file is not complete.
<code>gxCrashedPrintFileErr</code>	The spool file could not be opened.
<code>gxInvalidPrintFileVersion</code>	The version number of the spool file is not valid.

**SEE ALSO**

The `GXImageDocument` message is described on page 4-93.

**GXDespoolPage**

QuickDraw GX sends the `GXDespoolPage` message to obtain the next page to be printed from the spool file. If you perform your own spooling, you need to override the `GXDespoolPage` message to interpret your spool file format. Your override of the `GXDespoolPage` message must match the following formal declaration:

```
OSErr MyDespoolPage (gxSpoolFile aSpoolFile,
                    long pageNum, gxFormat aFormat,
                    gxShape *aShape, Boolean *formatChanged);
```

`aSpoolFile` The spool file.

`pageNum` The page number of the despoiled page.

`aFormat` A format object that is filled in by this function. This object needs to be preallocated.

`aShape` On return, a reference to the data that belongs on the page in the form of a picture shape.

`formatChanged`

On return, a Boolean value that is `true` if the page contains a format that is different from the previous format returned by the `GXDespoolPage` message and `false` if not.

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

**DESCRIPTION**

QuickDraw GX, in its default implementation of the `GXImagePage` message, sends the `GXDespoolPage` message to obtain the next page in the spool file to be printed.

You can override this message to change the default settings in a newly created format object. If you override the `GXSpoolPage` message to encrypt data, you need to override `GXDespoolPage` to decrypt the data.

## Printing Messages

The default implementation of the `GXDespoolPage` message reads the format and shape data for this page. It reads the page data from the spool file and repeatedly sends the `GXDespoolData` message.

**SPECIAL CONSIDERATIONS**

You rarely send the `GXDespoolPage` message yourself.

If you are providing your own spooling, you need to totally override the `GXDespoolPage` message and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward the `GXDespoolPage` message to allow the default implementation to read the data from the spool file.

Forward this message prior to modifying the graphics shape or related format object.

**RESULT CODES**

<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.
<code>gxIncompletePrintFileErr</code>	The spool file is not complete.
<code>gxCrashedPrintFileErr</code>	The spool file could not be opened.
<code>gxInvalidPrintFileVersion</code>	The version number of the spool file is not valid.

**SEE ALSO**

You can find an example of an override of the `GXDespoolPage` message in Listing 2-14 on page 2-25.

The `GXImagePage` message is described on page 4-94.

The `GXSpoolPage` message is described on page 4-69.

The `GXDespoolData` message is described in the next section.

**GXDespoolData**

---

QuickDraw GX sends the `GXDespoolData` message to read a stream of data from the spool file. If you perform your own spooling, you need to override the `GXDespoolData` message to interpret your spool file format. Your override of the `GXDespoolData` message must match the following formal declaration:

```
OSErr MyDespoolData (gxSpoolFile aSpoolFile,
                    Ptr data, long *length);
```

## Printing Messages

`aSpoolFile` The spool file.

`data` A pointer to the buffer that holds the data from the spool file.

`length` On entry, the length of the buffer in bytes. On return, the actual number of bytes that were despoiled.

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

**DESCRIPTION**

You can override the `GXDespoolData` message to decrypt data if you encrypted data in an override of the `GXSpoolData` message.

The default implementation of this message reads the requested amount of data from the spool file.

**SPECIAL CONSIDERATIONS**

You rarely send the `GXDespoolData` message yourself.

If you are providing your own spooling, you need to totally override the `GXDespoolData` message and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward the `GXDespoolData` message to allow the default implementation to read the data from the spool file. Forward this message prior to modifying the data.

**RESULT CODES**

<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.
<code>gxIncompletePrintFileErr</code>	The spool file is not complete.
<code>gxCrashedPrintFileErr</code>	The spool file could not be opened.
<code>gxInvalidPrintFileVersion</code>	The version number of the spool file is not valid.

**SEE ALSO**

The `GXSpoolData` message is described on page 4-70.

**GXDespoolResource**

QuickDraw GX sends the `GXDespoolResource` message to read a resource from the spool file. If you perform your own spooling, you need to override the

## Printing Messages

`GXDespoolResource` message to interpret your spool file format. Your override of the `GXDespoolResource` message must match the following formal declaration:

```
OSErr MyDespoolResource (gxSpoolFile aSpoolFile,
                        ResType aType, short id,
                        Handle *aResource);
```

`aSpoolFile` The spool file.

`aType` The resource type of the resource to read.

`id` The resource ID of the resource to read.

`aResource` On return, a handle to the resource that was despoiled.

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

## DESCRIPTION

You can override the `GXDespoolResource` message to decrypt a resource if you encrypted a resource in an override of the `GXSpoolResource` message.

The default implementation of this message reads the requested resource from the spool file.

## SPECIAL CONSIDERATIONS

You rarely send the `GXDespoolResource` message yourself.

If you are providing your own spooling, you need to totally override the `GXDespoolResource` message and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward the `GXDespoolResource` message to allow the default implementation to read the resource data from the spool file. Forward this message prior to modifying the resource data.

## RESULT CODES

<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.
<code>gxIncompletePrintFileErr</code>	The spool file is not complete.
<code>gxCrashedPrintFileErr</code>	The spool file could not be opened.
<code>gxInvalidPrintFileVersion</code>	The version number of the spool file is not valid.

## SEE ALSO

The `GXSpoolResource` is described on page 4-71.



## GXExamineSpoolFile

---

QuickDraw GX sends the `GXExamineSpoolFile` message just prior to establishing communications with a device. You need to override the `GXExamineSpoolFile` message if you have your own spool file format. Your override code of the `GXExamineSpoolFile` message must match the following formal declaration:

```
OSErr MyExamineSpoolFile (gxSpoolFile aSpoolFile);  
  
aSpoolFile  The spool file.
```

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

### DESCRIPTION

QuickDraw GX sends the `GXExamineSpoolFile` message just prior to opening the connection to the device.

If you are using the default spool-file format, you rarely need to override the `GXExamineSpoolFile` message.

The default implementation of this message checks the spool file for consistency before allowing it to be printed.

### SPECIAL CONSIDERATIONS

You never send the `GXExamineSpoolFile` message yourself. You rarely need to override this message.

If you are using your own spool file format, you need to totally override the `GXExamineSpoolFile` message. Otherwise, you need to forward it to other handlers, either before or after performing your own tasks.

### RESULT CODES

<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.

## GXCloseSpoolFile

---

QuickDraw GX sends the `GXCloseSpoolFile` message to close the spool file. If you perform your own spooling, you need to override the `GXCloseSpoolFile` message to

## Printing Messages

handle closing the spool file and updating it on disk. Your override of the `GXCloseSpoolFile` message must match the following formal declaration:

```
OSErr MyCloseSpoolFile (gxSpoolFile aSpoolFile,
                        long closeOptions);
```

`aSpoolFile`

The spool file.

`closeOptions`

Options for closing the spool file, as shown in Table 4-6.

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

## DESCRIPTION

QuickDraw GX sends the `GXCloseSpoolFile` message to close and possibly delete a spool file once imaging is complete. You can specify various options in the `closeOptions` parameter. The constants for the spool-file closing options are shown in Table 4-6.

**Table 4-6** `GXCloseSpoolFile` options

Constant	Value	Explanation
<code>gxNoCloseOptions</code>	<code>0x00000000</code>	No options are in effect
<code>gxDeleteOnClose</code>	<code>0x00000001</code>	The spool file is to be deleted rather than closed
<code>gxUpdateJobData</code>	<code>0x00000002</code>	QuickDraw GX needs to write the current job information into the spool file before closing it
<code>gxMakeRemoteFile</code>	<code>0x00000004</code>	QuickDraw GX needs to set the type of the spool file to 'rjob' rather than deleting it, which means that the file is sent to another machine for printing and that the original file is retained in case an error occurs

You can override the `GXCloseSpoolFile` message to perform any operations that you need to when the spool file is closed.

The default implementation of the `GXCloseSpoolFile` message includes the `gxDeleteOnClose` option.

## SPECIAL CONSIDERATIONS

You rarely send the `GXCloseSpoolFile` message yourself.

## Printing Messages

If you are providing your own spooling, you need to totally override the `GXCloseSpoolFile` message and all of the other spooling and despooling messages.

If you are not providing your own spooling, you must forward the `GXCloseSpoolFile` message to allow the default implementation to close the spool file.

**RESULT CODES**

<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.
<code>gxIncompletePrintFileErr</code>	The spool file is not complete.
<code>gxCrashedPrintFileErr</code>	The spool file could not be opened.
<code>gxInvalidPrintFileVersion</code>	The version number of the spool file is not valid.

**Dialog Box Messages**

You can use dialog box messages to communicate with users through dialog boxes. QuickDraw GX sends dialog box messages during the application phase of printing when the user interacts with the print dialog boxes. You can override these messages to add panels or modify the panels you present to the user.

**GXPrintingEvent**

QuickDraw GX sends the `GXPrintingEvent` message when events occur in a print dialog box. You can override the `GXPrintingEvent` message to handle events such as window update events that occur during display of print dialog boxes. Your override of the `GXPrintingEvent` message must match the following formal declaration:

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

`anEventRecord`

A pointer to an event that occurred in a print dialog box.

`filterEvent`

A Boolean value that is `true` if the event needs to be filtered, and `false` if not.

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

**DESCRIPTION**

QuickDraw GX sends the `GXPrintingEvent` message whenever a specific event occurs in one of the print dialog boxes that is displayed for printing. You can override this message if you need to process events that occur during printing.

## Printing Messages

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

**SPECIAL CONSIDERATIONS**

You never send the `GXPrintingEvent` message yourself.

You always create a total override of the `GXPrintingEvent` message.

**RESULT CODES**

<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.

**GXJobDefaultFormatDialog**

---

QuickDraw GX sends the `GXJobDefaultFormatDialog` message when the application displays the Page Setup dialog box. You can override the `GXJobDefaultFormatDialog` message to modify the behavior or appearance of the dialog box. Your override of the `GXJobDefaultFormatDialog` message must match the following formal declaration:

```
OSErr MyJobDefaultFormatDialog (gxDialogResult *aDialogResult);
```

`aDialogResult`

On return, a pointer to a value that specifies the selection made by the user in the dialog box.

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

**DESCRIPTION**

QuickDraw GX sends the `GXJobDefaultFormatDialog` message when the user clicks the More Choices button in the Page Setup dialog box. The application calls the `GXJobDefaultFormatDialog` function to display the extended Page Setup dialog box.

The default implementation of this message adds the standard printing panels and interface and then displays the dialog box.

You usually override this message to customize the dialog box by adding panels using the `GXSetupDialogPanel` function. You can add your own panels to the dialog box through the normal QuickDraw GX printing calls.

**SPECIAL CONSIDERATIONS**

You never send the `GXJobDefaultFormatDialog` message yourself.

You must forward the `GXJobDefaultFormatDialog` message to other message handlers. Add your panels and then forward the message.

**RESULT CODES**

<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**

You can find an example of an override of the `GXJobDefaultFormatDialog` message in Listing 3-7 on page 3-29 in the chapter “Printer Drivers.”

The `GXSetupDialogPanel` function is described on page 5-28 in the chapter “Printing Functions for Message Overrides.”

The `GXJobDefaultFormatDialog` function is described in *Inside Macintosh: QuickDraw GX Printing*.

## GXFormatDialog

---

QuickDraw GX sends the `GXFormatDialog` message when the application displays the Custom Page Setup dialog box. You can override the `GXFormatDialog` message to modify the behavior or appearance of the dialog box. Your override of the `GXFormatDialog` message must match the following formal declaration:

```
OSErr MyFormatDialog (gxFormat aFormat, StringPtr title,
                     gxDialogResult, *aDialogResult);
```

`aFormat`      A reference to the format object.

`title`        The title of the dialog box. If you specify `nil` as the value of this parameter, the title “Custom Page Setup” is used.

`aDialogResult`      On return, a pointer to the selection made by the user in the dialog box.

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

## Printing Messages

## DESCRIPTION

QuickDraw GX sends the `GXFormatDialog` message when the user selects the Custom Page Setup menu item and an application subsequently calls the `GXFormatDialog` function to display the Custom Page Setup dialog box.

The default implementation of this message adds the standard printing panels and interface and then displays the dialog box.

You usually override this message to customize the dialog box by adding panels using the `GXSetupDialogPanel` function.

## SPECIAL CONSIDERATIONS

You never send the `GXFormatDialog` message yourself.

You must forward the `GXFormatDialog` message to other message handlers. Add your panels and then forward the message.

## RESULT CODES

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

## SEE ALSO

The `GXFormatDialog` function is described in *Inside Macintosh: QuickDraw GX Printing*.

The `GXSetupDialogPanel` function is described on page 5-28 in the chapter “Printing Functions for Message Overrides.”

## GXJobPrintDialog

---

QuickDraw GX sends the `GXJobPrintDialog` message when the application displays the Print dialog box. You can override the `GXJobPrintDialog` message to modify the behavior or appearance of the Print dialog box. Your override of the `GXJobPrintDialog` message must match the following formal declaration:

```
OSErr MyJobPrintDialog (gxDialogResult *aDialogResult);
```

`aDialogResult`

On return, a pointer to the selection made by the user in the dialog box.

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

**DESCRIPTION**

QuickDraw GX sends the `GXJobPrintDialog` message when the user selects Print from the File menu and the application subsequently calls the `GXJobPrintDialog` function to display the Print dialog box on the user's screen.

The default implementation of this message adds the standard printing panels and interface and then displays the dialog box.

You usually override this message to customize the dialog box by adding panels using the `GXSetupDialogPanel` function.

**SPECIAL CONSIDERATIONS**

You never send the `GXJobPrintDialog` message yourself.

You must forward the `GXJobPrintDialog` message to other message handlers. Add your panels and then forward the message.

**RESULT CODES**

<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**

You can find an example of an override of the `GXJobPrintDialog` message in Listing 2-3 on page 2-13 in the chapter “Printing Extensions.”

The `GXSetupDialogPanel` function is described in the section “Adding a Panel to a Print Dialog Box” beginning on page 5-27 in the chapter “Printing Functions for Message Overrides.”

## **GXHandlePanelEvent**

---

QuickDraw GX sends the `GXHandlePanelEvent` message when an event happens in a panel. You can override the `GXHandlePanelEvent` message to handle panel events that cannot be handled using 'xdt1' resources. Your override of the `GXHandlePanelEvent` message must match the following formal declaration:

```
OSErr MyHandlePanelEvent (gxPanelInfoRecord *aPanelInfoRecord,
                          gxPanelResult *panelResult);
```

## Printing Messages

`aPanelInfoRecord`

A pointer to the panel information structure that supplies information to the panel about the current dialog box and panel event.

`panelResult`

On return, the result of handling the panel event. This is one of the values described in the section “Panel Responses” on page 4-37.

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

## DESCRIPTION

QuickDraw GX sends the `GXHandlePanelEvent` message to allow a panel to handle events associated with the dialog box.

The default implementation of this message does nothing. You need to override this message if you add panels that cannot be handled in a standard way (using 'xdt1' resources).

## SPECIAL CONSIDERATIONS

You never send the `GXHandlePanelEvent` message yourself.

You always perform a total override of the `GXHandlePanelEvent` message, in which you handle any events of interest that occur in your panel.

## RESULT CODES

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

## SEE ALSO

The panel information structure is described on page 4-35.

You can find an example of an override of the `GXHandlePanelEvent` message in Listing 2-12 on page 2-22 in the chapter “Printing Extensions.”

Print dialog boxes, panels, and the 'xdt1' resource are described in *Inside Macintosh: QuickDraw GX Printing*.

## GXFilterPanelEvent

---

QuickDraw GX sends the `GXFilterPanelEvent` message when an event happens in a panel. You can override the `GXFilterPanelEvent` message to add panels that need a



## Printing Messages

filter procedure. Your override of the `GXFilterPanelEvent` message must match the following formal declaration:

```
OSErr MyFilterPanelEvent (gxPanelInfoRecord *aPanelInfoRecord;
                          Boolean *returnImmed);
```

`aPanelInfoRecord`

A pointer to the panel information structure that supplies information to the panel about the current dialog box and panel event.

`returnImmed`

On return, a Boolean value that is `true` if there should be no further processing on this event and `false` if not.

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

**DESCRIPTION**

QuickDraw GX sends the `GXFilterPanelEvent` message to filter panel events in a dialog box.

The default implementation of this message does nothing. You need to override this message if you add panels that require a filtering process.

**SPECIAL CONSIDERATIONS**

You never send the `GXFilterPanelEvent` message yourself.

You always perform a total override of the `GXFilterPanelEvent` message, in which you filter any events that occur in your panel.

**RESULT CODES**

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

**SEE ALSO**

The panel information structure is described on page 4-35.

Print dialog boxes and panels are described in *Inside Macintosh: QuickDraw GX Printing*.

## Universal Imaging Messages

---

QuickDraw GX sends universal imaging messages during the imaging phase of printing to allow you to change a spool file into a format more readily accepted by the printer. In addition to these messages, you will see messages that are specific to each type of printer that the system supports: raster, PostScript, and vector.

## GXJobStatus

---

QuickDraw GX sends the `GXJobStatus` message to display the current status of a print job during spooling and despooling. You can override the `GXJobStatus` message to handle status at spooling and despooling times. Your override of the `GXJobStatus` message must match the following formal declaration:

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

`aStatusRecord`

A pointer to a status structure.

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

### DESCRIPTION

QuickDraw GX sends the `GXJobStatus` message when a printing extension or printer driver calls the `GXReportStatus` function.

An example of why you might override this message is to display the status in a window.

The default implementation of this message displays the status in the desktop printer window.

### SPECIAL CONSIDERATIONS

You never send the `GXJobStatus` message yourself.

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

### RESULT CODES

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

### SEE ALSO

The status structure is described in the section “The Status Structure” beginning on page 4-39.

The `GXReportStatus` function is described on page 5-17 in the chapter “Printing Functions for Message Overrides.”

## GXCaptureOutputDevice

---

QuickDraw GX sends the `GXCaptureOutputDevice` message to remove a device from a network or to reestablish its availability on the network. If you write drivers that communicate with network devices (other than PAP) that can be removed from the network, you can override the `GXCaptureOutputDevice` message to manage the capture and release of these devices. Your override of the `GXCaptureOutputDevice` message must match the following formal declaration:

```
OSErr MyCaptureOutputDevice (Boolean capture);
```

**capture**      A Boolean value that is `true` if you want the device captured and `false` if you want it released.

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

### DESCRIPTION

QuickDraw GX sends the `GXCaptureOutputDevice` message when it needs to remove a printing device from the network or to return the printing device to availability on the network.

The default implementation of the `GXCaptureOutputDevice` message automatically handles the capturing and restoring of PAP devices on a network. It uses a series of capture resources to control the capture process.

You can override this message if you write drivers that implement communications with network devices (other than PAP) that can be removed from the network. Your override implements the capture and release of these devices.

### SPECIAL CONSIDERATIONS

You never send the `GXCaptureOutputDevice` message yourself.

You always implement a total override of the `GXCaptureOutputDevice` message, in which you implement your own device capture and release strategy.

### RESULT CODES

<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

Capture resources are described in the section “The Capture ('cpts') Resource” beginning on page 6-63 in the chapter “Printing Resources.”

## GXImageJob

---

QuickDraw GX sends the `GXImageJob` message at the start of the imaging phase of printing, when a print job is ready to print. You can override the `GXImageJob` message to affect the way an entire job is printed. Your override of the `GXImageJob` message must match the following formal declaration:

```
OSErr MyImageJob (gxSpoolFile aSpoolFile, long *closeOptions);
```

`aSpoolFile` The spool file.

`closeOptions`

A pointer to the spool-file closing options for the job. These are passed on to the `GXCloseSpoolFile` message at the end of spooling. You can use the values that are shown in Table 4-6 on page 4-80.

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

### DESCRIPTION

QuickDraw GX sends the `GXImageJob` message at the start of imaging when it determines that a print job in the queue is ready to be printed.

The default implementation of `GXImageJob` prints an entire spool file. It first determines whether the file is to be printed on the local Macintosh or sent to a remote station. In the default implementation, QuickDraw GX sends the `GXOpenConnection` message to open the printer connection. Next, it sends the `GXSetupImageData` message so that the imaging system can set up its imaging time data. It also sends the `GXImageDocument` message to begin imaging the document. And finally, it sends the `GXCloseConnection` message to close the connection to the printer.

### SPECIAL CONSIDERATIONS

You never send the `GXImageJob` message yourself.

You must forward the `GXImageJob` message to other message handlers. You can perform your tasks before or after forwarding the message.

### RESULT CODES

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

## SEE ALSO

The `GXCloseSpoolFile` message is described on page 4-79.

The `GXSetupImageData` message is described on page 4-92.

The `GXImageDocument` message is described on page 4-93.

The `GXOpenConnection` message is described on page 4-131.

The `GXCloseConnection` message is described on page 4-135.

## GXCreateImageFile

---

QuickDraw GX sends the `GXCreateImageFile` message before the imaging of a print job begins. You need to override the `GXCreateImageFile` message if your device requires that an image file be created. Your override of the `GXCreateImageFile` message must match the following formal declaration:

```
OSErr MyCreateImageFile (FSSpecPtr aFSSpecPtr,
                        long imageFileOptions,
                        long *fileReference);
```

`aFSSpecPtr` Where to create the image file.

`imageFileOptions`

Options for creation of the image file, as shown in Table 4-7.

`fileReference`

On return, this value specifies a reference to the created image file.

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

## DESCRIPTION

You need to override the `GXCreateImageFile` message if you are working with an output device that needs to be driven from a file. For example, a film recorder might need to process all of the data for a single image plane at once, due to hardware timing constraints. Or a facsimile machine might need to receive the data for a page without lengthy interruptions that indicate a disconnection. For devices with constraints like these, you need to send the printer data to a file that is subsequently streamed to the device.

When you create an imaging file, QuickDraw GX stores the printing data and plays it back to the output device in a steady stream. The options that you specify in the `imageFileOptions` parameter define how the playback occurs, as shown in Table 4-7.

## Printing Messages

The default implementation of this message creates the image file if the options specify that it should.

**Table 4-7** Image file options

Constant	Value	Explanation
<code>gxNoImageFile</code>	0	The function does not create an image file. This is the default value.
<code>gxMakeImageFile</code>	1	The function creates an image file.
<code>gxEachPlane</code>	2	The function stores one image plane of data at a time. This allows a device like a film recorder to process each image plane of a data without pausing.
<code>gxEachPage</code>	4	The function stores one page of data at a time. This allows a device to process each page without pausing.
<code>gxEntireFile</code>	6	The function stores the entire document to allow the device to process the document without pausing.

## SPECIAL CONSIDERATIONS

You never send the `GXCreateImageFile` message yourself.

You must forward the `GXCreateImageFile` message to other message handlers. You can modify the options and then forward the message.

## RESULT CODES

<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.

## GXSetupImageData

QuickDraw GX sends the `GXSetupImageData` message to allow you to send initialization data that is specific to the kind of imaging system (raster, PostScript, or vector) that is being used for a job. You need to override the `GXSetupImageData` message if you want to modify imaging data. Your override of the `GXSetupImageData` message must match the following formal declaration:

```
OSErr MySetupImageData (void *imageData);
```

`imageData` A pointer to imaging-system-specific data for initializing the printing device.

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

#### DESCRIPTION

QuickDraw GX sends the `GXSetupImageData` message to initialize data specific to the type of printer that is being printed to: raster, PostScript, or vector.

The default implementation of this message reads the default imaging information from resources within the specific driver.

#### SPECIAL CONSIDERATIONS

You never send the `GXSetupImageData` message yourself.

You almost always forward the `GXSetupImageData` message to other message handlers. Forward the message to get the default values of the imaging data filled in, and then change the data as you require.

#### RESULT CODES

<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

You can find an example of an override of the `GXSetupImageData` message in Listing 3-11 on page 3-36 in the chapter “Printer Drivers.”

The raster imaging system structure, which contains the data used by the raster imaging system, is described on page 4-23.

The PostScript imaging system structure, which contains the data used by the PostScript imaging system, is described on page 4-26.

The vector imaging system structure, which contains the data used by the vector imaging system, is described on page 4-32.

## GXImageDocument

---

QuickDraw GX sends the `GXImageDocument` message just prior to starting the imaging of a document. You need to override the `GXImageDocument` message if you want to perform a task at the start of imaging for a document. Your override of the `GXImageDocument` message must match the following formal declaration:

```
OSErr MyImageDocument (gxSpoolFile aSpoolFile, void *imageData);
```

## Printing Messages

`aSpoolFile` The spool file to image.

`imageData` A pointer to imaging-system-specific data.

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

## DESCRIPTION

QuickDraw GX sends the `GXImageDocument` message to print a spool file.

The default implementation of the `GXImageDocument` message prints the document. First, it creates a new format to use when it calls the `GXDespoolPage` message. Then it sends the `GXCountPages` message to find out how many pages are on a spool file. It loops, sending the `GXImagePage` message for each page in the document. And finally, it disposes of the format that it allocated.

## SPECIAL CONSIDERATIONS

You never send the `GXImageDocument` message yourself.

You almost always forward the `GXImageDocument` message to other message handlers. You can forward the message before or after performing your own tasks.

## RESULT CODES

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

## SEE ALSO

The `GXDespoolPage` message is described on page 4-75.

The `GXCountPages` message is described on page 4-74.

The `GXImagePage` message is described on page 4-94.

## GXImagePage

---

QuickDraw GX sends the `GXImagePage` message just prior to starting the imaging of a page. You need to override the `GXImagePage` message if you want to perform some task for the imaging of each page. Your override of the `GXImagePage` message must match the following formal declaration:

```
OSErr MyImagePage (gxSpoolFile aSpoolFile, long pageNumber,
                  gxFormat aFormat, void *imageData);
```



## Printing Messages

`aSpoolFile` The file to print.  
`pageNumber` The page being imaged.  
`aFormat` The format object for the page.  
`imageData` A pointer to imaging-system-specific data.

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

## DESCRIPTION

QuickDraw GX, in its default implementation of the `GXImageDocument` message, sends the `GXImagePage` message once for each page in a spool file that is included in the count returned by the `GXCountPages` message.

The default implementation of this message prints the page. First, it sends the `GXStartSendPage` message to indicate that a page is being sent to a printer. Next, it sends the `GXRenderPage` message so that the imaging system can translate the page from graphics data into data for the specified printer. It then sends the `GXFinishSendPage` message to indicate that the page has been sent to the printer.

## SPECIAL CONSIDERATIONS

You never send the `GXImagePage` message yourself.

You almost always forward the `GXImagePage` message to other message handlers. You can forward the message before or after performing your own tasks.

## RESULT CODES

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

## SEE ALSO

The `GXImageDocument` message is described in the previous section.

The `GXCountPages` message is described on page 4-74.

The `GXStartSendPage` message is described on page 4-136.

The `GXRenderPage` message is described on page 4-96.

The `GXFinishSendPage` message is described on page 4-138.

## GXRenderPage

---

QuickDraw GX sends the `GXRenderPage` message just prior to the rendering the image of a page. You need to override the `GXRenderPage` message if you want to perform some task during rendering of each page. Your override of the `GXRenderPage` message must match the following formal declaration:

```
OSErr MyRenderPage (gxFormat aFormat, gxShape aShape,
                    gxPageInfoRecord *aPageInfoRecord,
                    void *imageData);
```

`aFormat`      The format object for the page.

`aShape`       The data that belongs on the page in the form of a graphics picture shape.

`aPageInfoRecord`

A pointer to a page information structure.

`imageData`   A pointer to imaging-system-specific data.

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

### DESCRIPTION

The default implementation of the `GXImagePage` message sends the `GXRenderPage` message once for each copy of a page. The `GXRenderPage` message renders a single page into raster, PostScript, or vector format.

You need to override this message if you want to perform some task prior to, during, or after the rendering of each page.

The default implementation of the `GXRenderPage` message renders one copy of the page, sending messages to relay that information to the specific driver.

### SPECIAL CONSIDERATIONS

You never send the `GXRenderPage` message yourself.

You almost always forward the `GXRenderPage` message to other message handlers. You can forward the message before or after performing your own tasks.

### RESULT CODES

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

### SEE ALSO

The page information structure is described on page 4-11.

The `GXImagePage` message is described in the previous section.

## Raster Imaging Messages

---

Raster imaging messages are sent only when the specific driver is imaging for a raster-based device. The type of device is selected when you specify raster imaging in the imaging system resource for your printing extension or printer driver. The imaging system ('isys') resource is described on page 6-33 in the chapter “Printing Resources.”

### GXRasterDataIn

---

You can override the `GXRasterDataIn` message to send data to a raster device for printing. Your override of the `GXRasterDataIn` message must match the following formal declaration:

```
OSErr MyRasterDataIn (gxOffscreenHdl offScreen,
                     gxRectangle *bandRectangle, gxRectangle *dirtyRectangle);
```

`offScreen` Data representing this portion of the page.

`bandRectangle`

A pointer to a rectangle that defines the size and location of the band of data that is being passed in.

`dirtyRectangle`

A pointer to a rectangle that defines the area of `bandRectangle` that is occupied by shapes.

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

#### DESCRIPTION

QuickDraw GX, in its default implementation of the `GXRenderPage` message for raster drivers, sends the `GXRasterDataIn` message after rendering a single band of data. The raster driver sends page data to the device in response to this message. Since a page cannot always be rendered as a single bitmap, QuickDraw GX can send this message multiple times for a page.

You can override this message to convert the bitmaps sent by QuickDraw GX into a format that is acceptable for your raster device. You can call the `GXBufferData` or `GXWriteData` messages to send the data to the device.

The default implementation of the `GXRasterDataIn` message breaks the data up into smaller pieces by subdividing the bitmap into “head passes” as defined in the raster package resource. It then sends `GXRasterLineFeed` and `GXRasterPackageBitmap` messages to further break down the data and prepare it for the device.

## Printing Messages

## SPECIAL CONSIDERATIONS

You never send the `GXRasterDataIn` message yourself.

You can totally override the `GXRasterDataIn` message, or you can modify the data in some way and then forward the message.

If you use the default implementation of the `GXRasterDataIn` message, you must define a raster package ('rpck') resource.

## RESULT CODES

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

## SEE ALSO

The raster package resource is described on page 6-73 in the chapter “Printing Resources.”

The `GXRenderPage` message is described on page 4-96.

The `GXRasterLineFeed` message is described on page 4-98.

The `GXRasterPackageBitmap` message is described on page 4-100.

The `GXBufferData` message is described on page 4-139.

The `GXWriteData` message is described on page 4-141.

## GXRasterLineFeed

---

You can override the `GXRasterLineFeed` message to send the codes to a printing device that cause it to move the print head. Your override of the `GXRasterLineFeed` message must match the following formal declaration:

```
OSErr MyRasterLineFeed (short *lineFeedSize,
                        Ptr buffer, unsigned long *bufferPos,
                        gxRasterImageDataHdl imageData);
```

`lineFeedSize`

The amount by which to move the print head.

`buffer`

A pointer to the buffer for creating the line-feed sequence.

`bufferPos`

The location in the buffer to create the sequence.

`imageData`

A pointer to raster imaging-system-specific data. This structure is described in the section “Raster Imaging System Structure” beginning on page 4-23.

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

#### DESCRIPTION

QuickDraw GX, in its default implementation of the `GXRasterDataIn` message, sends the `GXRasterLineFeed` message to move the print head up or down the page. This message creates a sequence of characters for the device to move the print head up or down the page, by an amount specified in the `lineFeedSize` parameter. The sequence of characters needs to be placed in the buffer (specified in the `bufferPos` parameter) from the beginning of the pointer.

The default implementation of this message sends the line-feed in a format specified in the raster package control resource for your driver. It uses specific strings defined in that resource to control the process of generating the sequence of characters from the requested line-feed size.

You can override this message to provide your own translation from the line-feed size into the sequence of characters. You override of the `GXRasterLineFeed` message needs to translate the line-feed size into a sequence of characters, store this sequence in the buffer, decrement the line-feed size by the amount translated, and increment the location in the buffer by the number of bytes added to the buffer.

#### Note

The value of the line-feed size may be any integer value (negative, zero, or positive). ♦

#### SPECIAL CONSIDERATIONS

You never send the `GXRasterLineFeed` message yourself.

You can totally override the `GXRasterLineFeed` message, or you can modify the data in some way and then forward the message.

If you use the default implementation of the `GXRasterLineFeed` message, you must define a raster package control (`'ropt'`) resource.

#### RESULT CODES

<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

You can find an example of an override of the `GXRasterLineFeed` message in Listing 3-3 on page 3-19 in the chapter “Printer Drivers.”

The raster package control resource is described on page 6-74 in the chapter “Printing Resources.”

## Printing Messages

The `GXRasterDataIn` message is described on page 4-97.

The raster imaging system structure is described on page 4-23.

## GXRasterPackageBitmap

---

QuickDraw GX sends the `GXRasterPackageBitmap` message to translate a portion of a bitmap into a sequence of characters for an output device to print. If you are writing a printing extension or printer driver, you can override the `GXRasterPackageBitmap` message to perform your own translation from bitmaps into character sequences for your output device. Your override of the `GXRasterPackageBitmap` message must match the following formal declaration:

```
OSErr MyRasterPackageBitmap (
                                gxRasterPackageBitmapRec *whattoPackage,
                                Ptr buffer, unsigned long *bufferPos,
                                gxRasterImageDataHdl imageData);
```

`whattoPackage`

The bitmap to translate and which part of it to package.

`buffer`

The buffer for creating the sequence of characters.

`bufferPos`

The location in the buffer to begin placing characters.

`imageData`

A pointer to raster imaging-system-specific data. This structure is described in the section “Raster Imaging System Structure” beginning on page 4-23.

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

### DESCRIPTION

QuickDraw GX, in its default implementation of the `GXRasterDataIn` message, sends the `GXRasterPackageBitmap` message to create a sequence of characters that represent a portion of the bitmap on the device. This message is sent multiple times per bitmap: once for each head pass that is required to print the bitmap, as specified in the raster package ('rpck') resource.

The default implementation of this message does nothing.

You always perform a total override of this message in your raster device driver to provide your own translation from bitmaps into a sequence of characters for the device. After translating part of the bitmap into a sequence of characters and storing this sequence into the buffer, you increment the buffer position (specified in the `bufferPos` parameter) by the number of bytes added to the buffer.

**SPECIAL CONSIDERATIONS**

You never send the `GXRasterPackageBitmap` message yourself.

You can partially override the `GXRasterPackageBitmap` message in an extension. In a raster device driver, you always totally override this message.

**RESULT CODES**

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

**SEE ALSO**

The `gxRasterPackageBitmapRec` data type is described in the section “Raster Package Bitmap Structure” on page 4-22.

The raster imaging system structure is described on page 4-23.

The raster package control resource type is described on page 6-74 in the chapter “Printing Resources.”

## PostScript Imaging Messages

---

QuickDraw GX sends PostScript imaging messages only when the specific driver is printing to a PostScript-based device. The type of device is selected when you specify PostScript imaging in the imaging system resource for your printing extension or printer driver. The imaging system ('`isys`') resource is described on page 6-33 in the chapter “Printing Resources.” The PostScript imaging messages are grouped in this section according to their functionality and you can use them for

- device configuration and control
- device communications
- management of procedure sets
- font management
- document-level structure and formatting control
- page-level structure and formatting control
- imaging control for shapes

## GXPostScriptQueryPrinter

---

QuickDraw GX sends the `GXPostScriptQueryPrinter` message to gather configuration information before imaging begins. You can override the `GXPostScriptQueryPrinter` message to gather information about the current

Printing Messages

state of the printer and to synchronize communications with it. Your override of the `GXPostScriptQueryPrinter` message must match the following formal declaration:

```
OSErr MyPostScriptQueryPrinter (long *queryResult);

queryResult
```

On return, specifies the state of the printer, as shown in Table 4-8.

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

DESCRIPTION

Ater the device connection is opened, QuickDraw GX sends the `GXPostScriptQueryPrinter` message. This message allows QuickDraw GX to gather information from a printer using two-way synchronous communications. The information gathered from the printer is stored in the desktop printer (DTP) configuration file, which all message handlers can access. The state of the printer is returned in the `queryResult` parameter. The values that are returned are shown in Table 4-8.

Table 4-8 Constants for PostScript query results

Constant	Value	Explanation
<code>gxPrinterOk</code>	0	The information gathered from the printer is sufficient to initiate imaging of the document. The information is stored in the configuration file.
<code>gxInitializePrinter</code>	1	The printer has not been initialized by any version of printing software and therefore must be initialized. After initialization, the information in the configuration file is up to date.
<code>gxFilePrinting</code>	2	The output of QuickDraw GX has been redirected to a file, and no immediate two-way synchronous communications is available with the printing device. QuickDraw GX checks that the required information exists in the configuration file and that it is adequate for imaging the document.
<code>gxResetPrinter</code>	128	The printer has been initialized with an incompatible version of printing software and therefore must be restarted and initialized. The information in the configuration file is left untouched.



The default implementation of this message verifies whether a communications channel is open to a printer or if the output of the system is being redirected to a file. If the print job is being sent to a printer, the default implementation of `GXPostScriptQueryPrinter` updates the printer configuration information by querying the printer to determine how much memory and which fonts are available on it. The outcome of this message is that the state of the PostScript imaging system is set up to print the job.

You can override the `GXPostScriptQueryPrinter` message to obtain information about the device you need to process the current document. You need to override this message if you want to test the state of an add-on hardware device such as a bin feeder.

#### SPECIAL CONSIDERATIONS

If you override the `GXPostScriptQueryPrinter` message, you must always forward it to enable the default implementation to return information about the type of connection.

#### RESULT CODES

<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.

The default implementation of the `GXPostScriptQueryPrinter` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

### **GXPostScriptInitializePrinter**

---

QuickDraw GX sends the `GXPostScriptInitializePrinter` message when the printer needs to be initialized. You can override the `GXPostScriptInitializePrinter` message to bring the printer to a known state after it has been restarted. Your override of the `GXPostScriptInitializePrinter` message must match the following formal declaration:

```
OSErr MyPostScriptInitializePrinter (void);
```

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

#### DESCRIPTION

QuickDraw GX sends the `GXPostScriptInitializePrinter` message when the `GXPostScriptQueryPrinter` message returns an indicator that a device needs to be initialized. The `GXPostScriptQueryPrinter` downloads any code necessary to initialize the printer to a known state.

## Printing Messages

The default implementation of this message downloads to the printer the latest version (version 7.0) of the LaserWriter PatchPrep procedure set, which defines a collection of PostScript operations that ensure compatibility between the LaserWriter driver and the printer. These operations then become “permanently” available on the printer, meaning that they are available until it is next restarted.

You can override this message if you want to modify the initialization process for the device. For example, you could modify the procedure set to add additional operations that are permanently available on the printer.

**SPECIAL CONSIDERATIONS**

Although you can totally override the `GXPostScriptInitializePrinter` message, you almost always forward it. Your implementation downloads any commands that you want permanently installed.

**RESULT CODES**

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

**SEE ALSO**

The `GXPostScriptQueryPrinter` message is described on page 4-101.

**GXPostScriptResetPrinter**

---

QuickDraw GX sends the `GXPostScriptResetPrinter` message when the printer needs to be reset. You can override the `GXPostScriptResetPrinter` message to alter the action taken when a printer is reset. Your override of the `GXPostScriptResetPrinter` message must match the following formal declaration:

```
OSErr MyPostScriptResetPrinter (void);
```

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

**DESCRIPTION**

QuickDraw GX calls the `GXPostScriptResetPrinter` message when the `GXPostScriptQueryPrinter` message indicates that a device requires resetting. The `GXPostScriptQueryPrinter` message forces a printer to restart itself, resetting it to a rebooted state.

You can override this message to replace the code that is used to reset the printer or to perform other actions when a printer is reset. The default implementation of this

message reboots the printer: it first sends the `GXPostScriptExitServer` message and then downloads PostScript code that executes the quit operation in the `systemdict` dictionary.

#### SPECIAL CONSIDERATIONS

You almost always forward the `GXPostScriptResetPrinter` message and then add your own reset actions. If you do totally override this message, you must make sure that your override reboots the printer state.

#### RESULT CODES

<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.

The default implementation of the `GXPostScriptResetPrinter` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

#### SEE ALSO

The `GXPostScriptExitServer` message is described in the next section.

The `GXPostScriptQueryPrinter` message is described on page 4-101.

### GXPostScriptExitServer

---

QuickDraw GX sends the `GXPostScriptExitServer` message when the printer needs to be restarted. You can override the `GXPostScriptExitServer` message to change the code used to exit the server loop in the printer. Your override of the `GXPostScriptExitServer` message must match the following formal declaration:

```
OSErr MyPostScriptExitServer (void);
```

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

#### DESCRIPTION

QuickDraw GX's default implementation of the `GXPostScriptQueryPrinter` message sends the `GXPostScriptExitServer` message when a device requires restarting. This message allows you to modify the permanent state of a printer.

When you exit the server loop, you can make changes that persist after the loop is reentered, such as changing the password. You must send an end-of-file (EOF) to the printer to reenter the server loop.

## Printing Messages

The default implementation of this message sends the following PostScript code:

```
serverdict begin 000000 exitserver
```

The 000000 value in this code is the default password for the printer.

**SPECIAL CONSIDERATIONS**

You can either partially or totally override the `GXPostScriptExit Server` message. If you override this message because the password has been changed, you must totally override it.

**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.

The default implementation of the `GXPostScriptExitServer` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

**SEE ALSO**

The `GXPostScriptQueryPrinter` message is described on page 4-101.

**GXPostScriptGetStatusText**

---

QuickDraw GX sends the `GXPostScriptGetStatusText` message to query the printer's status channel. You can override the `GXPostScriptGetStatusText` message to add your own handling to the receiving of status strings from the printer. Your override of the `GXPostScriptGetStatusText` message must match the following formal declaration:

```
OSErr MyPostScriptGetStatusText (Handle statusTextHdl);
```

```
statusTextHdl
```

A handle to the data that is output from the printer's status channel. The first long word in the data is treated as the length of the text.

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

**DESCRIPTION**

QuickDraw GX sends the `GXPostScriptGetStatusText` message to retrieve the status string from the printer's status channel. This message is sent by the default

implementation of the `GXOpenConnection` and `GXDumpBuffer` messages. It informs QuickDraw GX of the current state of a printer.

The default implementation of this message issues a `PAPStatus` call and uses the information stored in the desktop printer to find the device. If the connection is already opened, the default implementation uses the current AppleTalk connection.

QuickDraw GX allocates a handle to 512 bytes for the status information. You can resize this handle as needed to change the size. The format of the handle is as follows:

```
{
    long byteCount;
    char data[];
} **handle;
```

The value of the `byteCount` field must be less than or equal to the size of the handle.

#### SPECIAL CONSIDERATIONS

You can partially override the `GXPostScriptGetStatusText` message to add special handling to the default implementation. If you are not using a PAP connection, you must totally override this message.

#### RESULT CODES

<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.

The default implementation of the `GXPostScriptGetStatusText` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

#### SEE ALSO

The `GXOpenConnection` message is described on page 4-131.

The `GXDumpBuffer` message is described on page 4-142.

### GXPostScriptGetPrinterText

QuickDraw GX sends the `GXPostScriptGetPrinterText` message to query the printer's data channel and maintain asynchronous two-way communications with a printer. Your override of the `GXPostScriptGetPrinterText` message must match the following formal declaration:

```
OSErr MyPostScriptGetPrinterText (Handle printerTextHdl);
```

## Printing Messages

`printerTextHdl`

A handle to the data that is output from the printer's standard output channel. The first long word in the data is treated as the length of the text.

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

## DESCRIPTION

QuickDraw GX sends the `GXPostScriptGetPrinterText` message to retrieve any data that the printer sends back on the data channel. This message is sent by the default implementation of the `GXOpenConnection`, `GXFreeBuffer`, and `GXDumpBuffer` messages. It informs QuickDraw GX of the current state of a printer. This message is not sent when printed output is directed to a file.

The default implementation of `GXPostScriptGetPrinterText` message issues a `PAPRead` call on the currently opened AppleTalk connection. If any text is received, it is copied into the handle. If there is no text, the length word in the handle is set to 0.

QuickDraw GX allocates a handle to 512 bytes for the status information. You can resize this handle as needed to change the size. The format of the handle is as follows:

```
{
    long byteCount;
    char data[];
} **handle;
```

The value of the `byteCount` field must be less than or equal to the size of the handle.

## SPECIAL CONSIDERATIONS

You can partially override the `GXPostScriptGetPrinterText` message to add special handling to the default implementation. If you are not using a PAP connection, you must totally override this message.

## RESULT CODES

<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.

The default implementation of the `GXPostScriptGetPrinterText` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

## Printing Messages

## SEE ALSO

The `GXOpenConnection` message is described on page 4-131.

The `GXDumpBuffer` message is described on page 4-142.

The `GXFreeBuffer` message is described on page 4-143.

An example of using the `GXPostScriptGetPrinterText` and `GXPostScriptScanPrinterText` messages is shown in Listing 4-1.

**Listing 4-1** Using the `GXPostScriptGetPrinterText` and `GXPostScriptScanPrinterText` messages

```
Handle response;
/* download the request and flush the buffers */
anErr = DownloadResource(kPostScriptType, kPaperTrayQueryID);
nrequire(anErr, DownloadResource);

/* flush all buffers to the device */
anErr = Send_GXWriteData(nil, 0);
nrequire(anErr, FlushBuffers);

/* wait for a response from the printer */
response = NewHandleClear(sizeof(long));

anErr = MemError();
nrequire(anErr, GetResponseBuffer);

for (;;)
{
    if (anErr = Send_GXPostScriptGetPrinterText(response))
        break;
    if ((** long **) response) > 0
    {
        if (Munger(response, 4, "*", 1, "", 0) > 0)
            break;
        else
        {
            if (anErr=Send_GXPostScriptScanPrinterText(response))
                break;
        }
    }
}
```

## GXPostScriptScanStatusText

---

QuickDraw GX sends the `GXPostScriptScanStatusText` message after getting a printer status message back from the printer to interpret and prepare the status string. You can override the `GXPostScriptScanStatusText` message if the status string has information you need or if the format of this string is not standard and cannot be interpreted by QuickDraw GX's default implementation. Your override of the `GXPostScriptScanStatusText` message must match the following formal declaration:

```
OSErr MyPostScriptScanStatusText (Handle statusTextHdl);

statusTextHdl
    A handle to the text returned in the GXPostScriptGetStatusText
    message.
```

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

### DESCRIPTION

QuickDraw GX sends the `GXPostScriptScanStatusText` message every time any text is returned from the `GXPostScriptGetStatusText` message (that is, when the message does not return an empty buffer). The `GXPostScriptScanStatusText` message performs error detection and reporting, using the Postscript scan ('scan') resource to interpret the string. Normally, the only text returned by a printer is an error. This message searches the text buffers to determine the printing status and interprets the status string to determine the state of the printing process.

The default implementation of the `GXPostScriptScanStatusText` message searches the text string for special keywords that determine the status of the printing device. It also reformats the string for reading by a user and calls the `GXReportStatus` function to display the string in the desktop printer window.

You need to override this message if the status string holds information that is meaningful only to you or if the device to which the workstation is connected has a format different from that of Apple LaserWriters. Only the Apple LaserWriter format is understood by the default implementation of this message.

### SPECIAL CONSIDERATIONS

You can partially or totally override the `GXPostScriptScanStatusText` message.



**RESULT CODES**

<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.

The default implementation of the `GXPostScriptScanStatusText` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

**SEE ALSO**

The `GXPostScriptGetStatusText` message is described on page 4-106.

The `GXReportStatus` function is described on page 5-17 in the chapter “Printing Functions for Message Overrides.”

**GXPostScriptScanPrinterText**

---

QuickDraw GX sends the `GXPostScriptScanPrinterText` message after getting a printer data message back from the printer to interpret and prepare the printer string. You can override the `GXPostScriptScanPrinterText` message if the printer string contains information you wish to use or if the device to which the workstation is connected has a format different from that of Apple LaserWriters. Your override of the `GXPostScriptScanPrinterText` message must match the following formal declaration:

```
OSErr MyPostScriptScanPrinterText (Handle printerTextHdl);
```

```
printerTextHdl
```

A handle to the text returned in the `GXPostScriptGetPrinterText` message.

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

**DESCRIPTION**

QuickDraw GX sends the `GXPostScriptScanPrinterText` message every time any text is returned from the `GXPostScriptGetPrinterText` message (that is, when this message does not return an empty buffer). The `GXPostScriptScanPrinterText` message interprets the text of the printer string to determine the state of the printing process. It performs error detection and reporting, using the Postscript scan ('scan') resource to interpret the string.

The default implementation of this message searches the text string for special keywords that determine the status of the printing device. It also reformats the string for reading by a user and calls the `GXReportStatus` function to display the string in the desktop printer window.

## Printing Messages

You need to override this message when part or all of the printer string holds some piece of information that is only meaningful to you or if the device to which the workstation is connected has a format different from that of Apple LaserWriters. Only the Apple LaserWriter format is understood by the default implementation of this message.

**SPECIAL CONSIDERATIONS**

You can partially or totally override the `GXPostScriptScanPrinterText` message.

**RESULT CODES**

<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.

The default implementation of the `GXPostScriptScanPrinterText` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

**SEE ALSO**

The `GXPostScriptGetPrinterText` message is described on page 4-107.

An example of using the `GXPostScriptGetPrinterText` and `GXPostScriptScanPrinterText` messages is shown in Listing 4-1 on page 4-109.

The `GXReportStatus` function is described on page 5-17 in the chapter “Printing Functions for Message Overrides.”

## **GXPostScriptGetDocumentProcSetList**

---

QuickDraw GX sends the `GXPostScriptGetDocumentProcSetList` message at the start of imaging for a document to determine which procedure sets are needed for the document. You can override the `GXPostScriptGetDocumentProcSetList` message to retrieve information about those procedure sets. Your override of the `GXPostScriptGetDocumentProcSetList` message must match the following formal declaration:

```
OSErr MyPostScriptGetDocumentProcSetList (
    gxProcSetListHdl procSetListHdl,
    gxPostScriptImageDataHandle hImageData);
```

## Printing Messages

`procSetListHdl`

On entry, a handle to a preallocated PostScript procedure set list structure.  
On return, a value that specifies the procedure sets needed to image the document.

`hImageData`

A handle to the PostScript imaging system structure.

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

**DESCRIPTION**

QuickDraw GX sends the `GXPostScriptGetDocumentProcSetList` message when you send the `GXImageDocument` message. The `GXPostScriptGetDocumentProcSetList` message gathers information on which procedure sets are needed to image a specified document.

The default implementation of this message returns the procedure sets needed by the PostScript generic driver, the PostScript imaging engine, and the font handler. The set of procedure sets for all documents are typically the same: those that are specified in the imaging system's structure, as determined when the printer is initially queried. However, some documents might require special handling, which means additional procedure sets need to be downloaded.

You can then use the `GXPostScriptDownloadProcSetList` message to make these procedures available in the PostScript device.

**SPECIAL CONSIDERATIONS**

You must forward the `GXPostScriptGetDocumentProcSetList` message.

**RESULT CODES**

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

**SEE ALSO**

The PostScript procedure set list structure is described on page 4-29.

The PostScript imaging system structure is described on page 4-26.

The `GXImageDocument` message is described on page 4-93.

The `GXPostScriptDownloadProcSetList` message is described in the next section.

An example of overriding the `GXPostScriptGetDocumentProcSetList` message is shown in Listing 4-2. This example downloads a procedure set that corrects a known bug in the PostScript implementation in the LaserWriter IIg and IIx printers.

**Listing 4-2** An example of the GXPostScriptGetDocumentProcSetList message

---

```

OSErr DriverGetDocumentProcSetList (gxProcSetListHdl procSetList,
                                   gxPostScriptImageDataHdl imageDataHdl)
{
    OSErr anErr;

    anErr = Forward_GXPostScriptGetDocumentProcSetList(
                                   procSetList, imageDataHdl);

    if (anErr == noErr)
    {
        /*
         * Send this proc set to the IIg or IIIf, or for portable
         * PostScript.
         */
        if
        (
            ((*imageDataHdl).renderOptions &
             gxPortablePostScriptOption) ||
            (gPrinterType == kLWIIIf) ||
            (gPrinterType == kLWIIg)
        )
        {
            SetHandleSize( (Handle) procSetList, GetHandleSize(
                (Handle) procSetList) + sizeof(ProcSetListRec));
            anErr = MemError();
            if (anErr == noErr)
            {
                gxProcSetListPtr pList = (gxProcSetListPtr)
                    ((unsigned long) (*procSetList)
                     + GetHandleSize(Handle) procSetList)
                    - sizeof(gxProcSetListRec);
                pList->clientid = 'drvr';
                pList->controlType = gxPostscriptProcSetControlType;
                pList->controlid = kFAndGShowPatch;
                pList->dataType = 'rdws';
            }
        }
    }
    return(anErr);
}

```

## GXPostScriptDownloadProcSetList

---

QuickDraw GX sends the `GXPostScriptDownloadProcSetList` message to guarantee that the needed procedure sets are available on the printer. You can override the `GXPostScriptDownloadProcSetList` message to modify the list of procedure sets that are needed to print a document. Your override of the `GXPostScriptDownloadProcSetList` message must match the following formal declaration:

```
OSErr MyPostScriptDownloadProcSetList (
                                gxProcSetListHdl procSetListHdl,
                                gxPostScriptImageDataHandle hImageData);
```

`procSetListHdl`

A handle to a PostScript procedure set list structure. This handle is filled by the `GXPostScriptGetDocumentProcSetList` message.

`hImageData`

A handle to the PostScript imaging system structure.

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

### DESCRIPTION

QuickDraw GX sends the `GXPostScriptDownloadProcSetList` message to guarantee that each of the procedure sets in the procedure set list is available in the printer. QuickDraw GX first sends the `GXPostScriptGetDocumentProcSetList` message to determine which procedure sets are needed for the document, and then sends this message to make sure that the procedure sets are downloaded to the printer.

The default implementation of the `GXPostScriptDownloadProcSetList` message downloads each of the procedure sets to the printer. You can override this message if you want to change the list of procedure sets that are going to be downloaded to the printer.

### SPECIAL CONSIDERATIONS

You must forward the `GXPostScriptDownloadProcSetList` message.

### RESULT CODES

<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.

The default implementation of the `GXPostScriptDownloadProcSetList` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

## SEE ALSO

The PostScript procedure set list structure is described on page 4-29.

The PostScript imaging system structure is described on page 4-26.

The `GXPostScriptGetDocumentProcSetList` message is described in the previous section.

## GXPostScriptGetPrinterGlyphsInformation

---

QuickDraw GX sends the `GXPostScriptGetPrinterGlyphsInformation` message to allow a printing extension or printer driver to communicate with the imaging system about the fonts and glyphs that are resident in the output device. You can override the `GXPostScriptGetPrinterGlyphsInformation` message if you are doing your own font management. Your override of the `GXPostScriptGetPrinterGlyphsInformation` message must match the following formal declaration:

```
OSErr MyPostScriptGetPrinterGlyphsInformation (
                                gxPrinterGlyphsRec *glyphPtr);
```

`glyphPtr`     A pointer to a PostScript glyphs structure.

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

## DESCRIPTION

The PostScript imaging system sends the `GXPostScriptGetPrinterGlyphsInformation` message before it starts to image a document. It uses the information gathered by this message to determine which fonts or which glyphs in a font need to be downloaded to the printer for the document to be printed.

On entry to this message, the `theFont` field of the PostScript glyphs structure, of data type `gxPrinterGlyphsRec`, must be filled in with a valid font reference to the font for which information is desired. You can fill in the `platform`, `script`, and `language` fields of the PostScript glyphs structure to tell the imaging system which glyphs from a font are present in the printer.

If the `platform` field of the structure has the value `-1`, then the `script` and `language` field values are ignored, and the `glyphBits` array is filled in with the glyphs that are actually present for the font.

If the `platform` field of the structure has any value other than `-1`, then the `platform`, `script`, and `language` fields together define how the imaging system maps glyphs into the printer's encoding scheme.

The default implementation of the `GXPostScriptGetPrinterGlyphsInformation` message uses the information that was gathered by the `GXPostScriptQueryPrinter` message to fill out the `PostScript glyphs` structure. It also looks for a 'pfont' resource that has the same font name as that in the `theFont` field of the structure and gathers information from that resource.

#### SPECIAL CONSIDERATIONS

You must forward the `GXPostScriptGetPrinterGlyphsInformation` message.

#### RESULT CODES

<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.

The default implementation of the `GXPostScriptGetPrinterGlyphsInformation` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

#### SEE ALSO

The `PostScript glyphs` structure is described on page 4-28.

The `GXPostScriptQueryPrinter` messages is described on page 4-101.

The `PostScript printer font ('pfont')` resource is described on page 6-84 in the chapter "Printing Resources."

For more information about how the imaging system maps glyphs, read about the 'cmap' table in the book *Inside Macintosh: QuickDraw GX Typography*.

## GXPostScriptStreamFont

QuickDraw GX sends the `GXPostScriptStreamFont` message when the imaging system determines that a font is needed to print a document. You can override the `GXPostScriptStreamFont` message to provide font management on a remote host. Your override of the `PostScriptStreamFont` message must match the following formal declaration:

```
OSErr MyPostScriptStreamFont (gxFont fontRef,
                              gxScalerStream *stream);
```

`fontRef`      A reference to the font to be streamed.

`stream`      A pointer to the `gxScalerStream` structure that specifies how to stream the font to the printer.

## Printing Messages

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

**DESCRIPTION**

The default implementation of the `GXPostScriptStreamFont` message streams the specified font to the printer. You can override this message to determine when a font is being downloaded or to change the streaming method used to send the font to the printer.

**SPECIAL CONSIDERATIONS**

You must forward the `GXPostScriptStreamFont` message.

**RESULT CODES**

<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.

The default implementation of the `GXPostScriptStreamFont` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

**SEE ALSO**

The `gxScalerStream` structure is described in *Inside Macintosh: QuickDraw GX Environment and Utilities*.

## GXPostScriptDoDocumentHeader

---

QuickDraw GX sends the `GXPostScriptDoDocumentHeader` message to generate the PostScript program header for a document. You can override the `GXPostScriptDoDocumentHeader` message to change the PostScript document structure and formatting setup for a document. Your override of the `GXPostScriptDoDocumentHeader` message must match the following formal declaration:

```
OSErr MyPostScriptDoDocumentHeader (
                                gxPostScriptImageDataHandle hImageData);
```

`hImageData`

A handle to the PostScript imaging system structure.

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



**DESCRIPTION**

QuickDraw GX sends the `GXPostScriptDoDocumentHeader` message during its default implementation of the `GXImageDocument` message when printing to a PostScript device. The `GXPostScriptDoDocumentHeader` message is the first message sent when generating a PostScript program. QuickDraw GX uses this message to generate a PostScript program header, which is required for conforming PostScript programs. The information for the header is obtained from the job object.

You can override this message to build your own document header or to add comments to a PostScript header.

**SPECIAL CONSIDERATIONS**

You can totally override the `GXPostScriptDoDocumentHeader` message to build your own PostScript program header, or you can forward this message and then add your own information to the header created by the default implementation.

**RESULT CODES**

<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.

The default implementation of the `GXPostScriptDoDocumentHeader` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

**SEE ALSO**

The `GXImageDocument` message is described on page 4-93.

The PostScript imaging system structure is described on page 4-26.

## **GXPostScriptDoDocumentSetup**

---

QuickDraw GX sends the `GXPostScriptDoDocumentSetup` message to set up document-level formatting information. You can override the `GXPostScriptDoDocumentSetup` message to issue any code that is required for the imaging of the whole document. Your override of the `GXPostScriptDoDocumentSetup` message must match the following formal declaration:

```
OSErr MyPostScriptDoDocumentSetup (
                                gxPostScriptImageDataHandle hImageData);
```

`hImageData`

A handle to the PostScript imaging system structure.

## Printing Messages

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

**DESCRIPTION**

QuickDraw GX sends the `GXPostScriptDoDocumentSetup` message in its default implementation of the `GXImageDocument` message, prior to imaging the first page of the document. You can override this message to issue any PostScript instructions that apply to the document; for example, to set up the default halftone accuracy.

The default implementation of this message changes the display in the desktop printer window to reflect the current document and user names.

**SPECIAL CONSIDERATIONS**

You always forward the `GXPostScriptDoDocumentSetup` message to allow other message handlers the opportunity to perform document-setup tasks. Forward this message and then add your own operations.

**RESULT CODES**

<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.

The default implementation of the `GXPostScriptDoDocumentSetup` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

**SEE ALSO**

The PostScript imaging system structure is described on page 4-26.

The `GXImageDocument` message is described on page 4-93.

**GXPostScriptDoDocumentTrailer**

---

QuickDraw GX sends the `GXPostScriptDoDocumentTrailer` message to generate the document trailer that is appended to the end of the PostScript program for a document. You can override the `GXPostScriptDoDocumentTrailer` message to perform an action after all pages in a document have been imaged. Your override of the `GXPostScriptDoDocumentTrailer` message must match the following formal declaration:

```
OSErr MyPostScriptDoDocumentTrailer (
                                gxPostScriptImageDataHandle hImageData);
```

`hImageData` A handle to the PostScript imaging system structure.

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

#### DESCRIPTION

QuickDraw GX sends the `GXPostScriptDoDocumentTrailer` message to generate the document trailer after terminating the imaging of a document. It sends this message during its default implementation of the `GXImageDocument` message, after all the pages of the document have been imaged.

The default implementation of this message changes the display in the desktop printer window to reflect the number of pages printed for the document.

#### SPECIAL CONSIDERATIONS

You always forward the `GXPostScriptDoDocumentTrailer` message to allow other message handlers the opportunity to perform document-termination tasks.

#### RESULT CODES

<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.

The default implementation of the `GXPostScriptDoDocumentTrailer` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

#### SEE ALSO

The `GXImageDocument` message is described on page 4-93.

The PostScript imaging system structure is described on page 4-26.

### **GXPostScriptDoPageSetup**

---

QuickDraw GX sends the `GXPostScriptDoPageSetup` message to set up page-level formatting information. You can override the `GXPostScriptDoPageSetup` message to customize the page setup information. Your override of the `GXPostScriptDoPageSetup` message must match the following formal declaration:

```
OSErr MyPostScriptDoPageSetup (gxFormat aFormat, long pageIndex,
                                gxPostScriptImageDataHandle hImageData);
```

`aFormat`      The format object for the page to be printed.

`pageIndex`    The number of the page that is about to be imaged.

## Printing Messages

`hImageData` A handle to the PostScript imaging system structure.

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

**DESCRIPTION**

QuickDraw GX sends the `GXPostScriptDoPageSetup` message after a page has been spooled and before it is imaged. This message performs all necessary actions for setting up the page in the printer, including initiating the `GXPostScriptSelectPaperType` message.

The default implementation of this message takes the page-formatting information and translates it into PostScript code. You can override this message if you need to perform any special actions for specific page numbers.

**SPECIAL CONSIDERATIONS**

You always forward the `GXPostScriptDoPageSetup` message to allow other message handlers the opportunity to perform page setup tasks.

**RESULT CODES**

<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.

The default implementation of the `GXPostScriptDoPageSetup` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

**SEE ALSO**

The PostScript imaging system structure is described on page 4-26.

The `GXPostScriptSelectPaperType` message is described in the next section.

## **GXPostScriptSelectPaperType**

---

QuickDraw GX sends the `GXPostScriptSelectPaperType` message prior to imaging the page, to select the printer's paper type for the page. You can override the `GXPostScriptSelectPaperType` message to do anything you need to do to make sure the device is set up correctly. Your override of the `GXPostScriptSelectPaperType` message must match the following formal declaration:

## Printing Messages

```
OSErr MyPostScriptSelectPaperType (gxPaperType aPaperType,
                                   long pageIndex, gxPostScriptImageDataHandle hImageData);
```

**aPaperType** The paper-type object that is needed.

**pageIndex** The page number of the page that is about to be imaged.

**hImageData** A handle to the PostScript imaging system structure.

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

**DESCRIPTION**

The PostScript generic driver invokes the `GXPostScriptSelectPaperType` message when it is setting up the format for a page—after the page has been despoiled but before it is imaged. This message selects the paper type for a printer. This message is sent by the default implementation of the `GXPostScriptDoPageSetup` message.

The default implementation of this message looks for a collection item of type 'post' in the paper-type collection and sends that to the printer as the paper type for the page. The collection item must be valid PostScript that works for all PostScript devices.

You can override this message to customize the way that setting the paper type for the page works.

**SPECIAL CONSIDERATIONS**

You usually forward the `GXPostScriptSelectPaperType` message to other message handlers; however, if you are overriding this message to issue your own `setpagedevice` operation, do not forward this message.

**RESULT CODES**

<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.
<code>gxPaperTypeNotFound</code>	The specified paper type could not be found.
<code>gxNoSuchPTGroup</code>	The specified paper type could not be found.

The default implementation of the `GXPostScriptSelectPaperType` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

**SEE ALSO**

The PostScript imaging system structure is described on page 4-26.

Paper types and paper-type collection items are described in *Inside Macintosh: QuickDraw GX Printing*.

## GXPostScriptDoPageTrailer

---

QuickDraw GX sends the `GXPostScriptDoPageTrailer` message to generate the page trailer that is appended to the end of the PostScript program for a page. You can override the `GXPostScriptDoPageTrailer` message if you need to add comments to the page trailer. Your override of the `GXPostScriptDoPageTrailer` message must match the following formal declaration:

```
OSErr MyPostScriptDoPageTrailer (
                                gxPostScriptImageDataHandle hImageData);
```

`hImageData` A handle to the PostScript imaging system structure.

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

### DESCRIPTION

QuickDraw GX sends the `GXPostScriptDoPageTrailer` message at the end of every page to provide any features requiring control.

The default implementation of the `GXPostScriptDoPageTrailer` message sends the `GXPostScriptEjectPage` message to eject a page. You can override this message to create a PostScript page trailer.

### SPECIAL CONSIDERATIONS

You usually forward the `GXPostScriptDoPageTrailer` message to allow other message handlers the opportunity to perform page-termination tasks and so that the default implementation can perform its operations.

### RESULT CODES

<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.

The default implementation of the `GXPostScriptDoPageTrailer` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

### SEE ALSO

The PostScript imaging system structure is described on page 4-26.

The `GXPostScriptEjectPage` message is described in the next section.

## GXPostScriptEjectPage

---

QuickDraw GX sends the `GXPostScriptEjectPage` message when printing of a page is done and the printer is ready for the next page. You can override the `GXPostScriptEjectPage` message to perform any actions required after a page has been printed and before the next page begins. Your override of the `GXPostScriptEjectPage` message must match the following formal declaration:

```
OSErr MyPostScriptEjectPage (gxPaperType aPaperType,
                             long pageIndex, long copiesCount, short erasePage,
                             gxPostScriptImageDataHandle hImageData);
```

`aPaperType` The paper-type object for the page that was just printed.

`pageIndex` The page number of the page that was just printed.

`copiesCount` The number of copies of the page needed.

`erasePage` A value that indicates whether the printer's frame buffer needs to be erased when the page is ejected.

`hImageData` A handle to the PostScript imaging system structure.

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

### DESCRIPTION

QuickDraw GX sends the `GXPostScriptEjectPage` message when the imaging system has completed imaging tasks for a single page and is ready for the next page.

You can specify whether the page image is to be erased from the printer's frame buffer in the `erasePage` parameter. If this value is `true`, the image is erased from the page buffer. If the difference between the page that was just printed and the next page is minimal, not erasing the frame buffer can significantly improve the printing speed. This is often used when printing forms.

The default implementation of this message uses the PostScript `showpage` operator when the page image needs to be erased and uses the PostScript `copypage` operator when the page image is not to be erased.

### SPECIAL CONSIDERATIONS

You usually forward the `GXPostScriptEjectPage` message to allow other message handlers the opportunity to determine if the frame buffer needs to be erased.

## Printing Messages

## 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.

The default implementation of the `GXPostScriptEjectPage` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

## SEE ALSO

The PostScript imaging system structure is described on page 4-26.

## GXPostScriptProcessShape

---

QuickDraw GX sends the `GXPostScriptProcessShape` message when converting a shape object into PostScript. You can override the `GXPostScriptProcessShape` message to exercise imaging control for a shape. Your override of the `GXPostScriptProcessShape` message must match the following formal declaration:

```
OSErr MyPostScriptProcessShape (gxShape aShape, long count,
                                gxTransform list[]);
```

<code>aShape</code>	The shape object that is to be printed.
<code>count</code>	The number of transform objects in the transform list.
<code>list</code>	A list of transforms through which the shape needs to be mapped when converted to PostScript.

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

## DESCRIPTION

QuickDraw GX sends the `GXPostScriptProcessShape` message to provide you with imaging control for individual shapes.

The default implementation of this message takes a shape and produces the PostScript code needed to image the shape.

You can override this message to change the way that a shape is imaged on a PostScript device.

**Note**

QuickDraw GX internally tracks the state of the printer. If you override this message, then QuickDraw GX's version of the printer's state can become out of sync with the hardware. This means that if you override this message for any shapes, you need to override it for all shapes. ♦



**SPECIAL CONSIDERATIONS**

You can modify the shape or the transform list before forwarding the `GXPostScriptProcessShape` message, or you can totally override the `GXPostScriptProcessShape` message to change the PostScript code that is generated for the shape.

**RESULT CODES**

<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.

**Vector Imaging Messages**

---

QuickDraw GX sends vector imaging messages only when the specific driver is printing for a vector-based device. The type of device is selected using the imaging system ('isys') resource, which is described on page 6-33 in the chapter “Printing Resources.”

**GXVectorPackageShape**

---

QuickDraw GX sends the `GXVectorPackageShape` message to translate a shape into a package that is sent to the printing device. You need to override the `GXVectorPackageShape` message to turn a shape into the appropriate pen commands. Your override of the `GXVectorPackageShape` message must match the following formal declaration:

```
OSErr MyVectorPackageShape (gxShape aShape, long penIndex);
```

`aShape`        The shape object to be packaged.  
`penIndex`      The index of the pen to use to draw the shape.

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

**DESCRIPTION**

The default implementation of the `GXRenderPage` message for the vector imaging system sends the `GXVectorPackageShape` message to translate a QuickDraw GX shape into a package and send it to a printing device.

The default implementation of this message does nothing. Your override needs to convert the shape into a series of pen commands and send those commands to your printing device with the `GXBufferData` or `GXWriteData` messages.

## Printing Messages

QuickDraw GX converts high-level shapes such as bitmaps, paths, curves, glyphs, text, and text layout into lower-level vector-type shapes such as polygons, rectangles, and lines. For all shapes, QuickDraw GX color sorts, applies transfer modes if needed, clips them against overlapping shapes, and resolves styles, inks, and transforms. It converts the resultant shape into a lower-level shape, if necessary, and then sends the `GXVectorPackageShape` message.

Your override of the `GXVectorPackageShape` message needs to convert the shape sent by QuickDraw GX into a device-specific language such as HPGL. You can also override this message to perform other tasks, such as supporting a preview feature in which data is displayed on the user's screen and written to a file before, or instead of, being sent to the device.

**SPECIAL CONSIDERATIONS**

You never send the `GXVectorPackageShape` message yourself.

You always totally override the `GXVectorPackageShape` message.

**RESULT CODES**

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

**SEE ALSO**

The `GXRenderPage` message is described on page 4-96.

The `GXBufferData` message is described on page 4-139.

The `GXWriteData` message is described on page 4-141.

**GXVectorLoadPens**

---

QuickDraw GX sends the `GXVectorLoadPens` message when drawing with the pens currently in the carousel is finished and it is time to move on to the next carousel. You can override the `GXVectorLoadPens` message to inform QuickDraw GX of the number and position of pens in the plotter. Your override of the `GXVectorLoadPens` message must match the following formal declaration:

```
OSErr MyVectorLoadPens (gxPenTableHdl penTable,
                        long *shapeCounts, Boolean *penTableChanged);
```

`penTable`     A handle to the pen table. This structure is described in the section “Vector Pen Table Structure” beginning on page 4-34.

## Printing Messages

shapeCounts

A pointer to an array of shape counts that corresponds to the pens in the pen table.

\*penTableChanged

A pointer to a Boolean value. On return, the value is `true` if the pen table is modified and `false` if it remains the same.

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

## DESCRIPTION

QuickDraw GX's default implementation of the `GXRenderPage` message for the vector imaging system sends the `GXVectorLoadPens` message when all imaging has taken place for the pens in the current pen carousel and it is time to move on to the next carousel. You can use the `GXVectorLoadPens` message to prompt the user to change pens loaded in the carousel.

The default implementation of this message does nothing. Your override of `GXVectorLoadPens` needs to mark all of the currently loaded pens as not loaded and prompt the user to install the new pen carousel. You then update the positions of the pens in the pen table to reflect their positions in the plotter. If you do not override this message, the number of pens in the pen table must be the same as what is currently loaded in the carousel.

**Note**

The `GXVectorLoadPens` message should only update the pen position field of the pens. It should not rearrange pens or remove, add, or change any other fields of the pen table. ♦

## SPECIAL CONSIDERATIONS

You never send the `GXVectorLoadPens` message yourself.

You always totally override the `GXVectorLoadPens` 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.
<code>gxInvalidPenTable</code>	The specified pen table is not valid.

## SEE ALSO

The `GXRenderPage` message is described on page 4-96.

The vector pen table structure is described on page 4-34.

## GXVectorVectorizeShape

---

QuickDraw GX sends the `GXVectorVectorizeShape` message to convert a complex shape into simpler, device-level shapes such as polygons. You can override the `GXVectorVectorizeShape` message to change the plotting of particular shapes. Your override of the `GXVectorVectorizeShape` message must match the following formal declaration:

```
OSErr MyVectorVectorizeShape (gxShape aShape, long penIndex,
                              gxVectorShapeDataRec *aVectorShapeDataRec);
```

`aShape`        The shape object to be vectorized.

`penIndex`     The index of the pen to draw the shape.

`aVectorShapeDataRec`

On return, a pointer to a vector shape structure that contains the vector data created from the shape.

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

### DESCRIPTION

QuickDraw GX's default implementation of `GXRenderPage` sends the `GXVectorVectorizeShape` message when a shape is ready to be converted into simpler device-level shapes. It sends the message to vectorize a specified shape (convert it into line segments). It sends the message after all transfer modes are resolved and all clipping and transforms have been concatenated.

QuickDraw GX's default implementation of this message converts shapes into polygons.

You can override this message to effect the plotting of particular shape types. You may find this useful for plotters that can handle certain shapes more easily than the shapes provided by the default implementation. You can completely remove undesired shapes, handle certain shapes entirely, modify shapes prior to forwarding the message, and send some shapes through unmodified.

The `GXVectorVectorizeShape` message is initiated before any style mappings are applied to the shape and before it is converted into a low-level shape. The message is invoked after clipping, color sorting, and transfer-mode resolution are complete. Before the message is sent, all the parent clips and mappings are concatenated with the shape's clip and mapping. The shape is left as close to its original form as possible. For example, text remains text shape and is not converted into a path.

You can override this message to customize rendering of certain shapes on your vector device. For example, if you prefer to use the device's native fonts to plot text, glyph, and layout shapes, you can override this message. Note that it is your responsibility to apply style and transforms to the shape.

**SPECIAL CONSIDERATIONS**

You never send the `GXVectorVectorizeShape` message yourself.

You can totally override the `GXVectorVectorizeShape` message, or you can partially override it to modify the default processing of certain shape types.

**RESULT CODES**

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

**SEE ALSO**

The `GXRenderPage` message is described on page 4-96.

The vector shape structure is described on page 4-31.

## Device Communications Messages

---

QuickDraw GX sends device communications messages to send data to a device during the device communications phase of printing. It is very important to note that only in your overrides of these messages can you actually communicate with the device or report problems about the device to the user.

## GXOpenConnection

---

QuickDraw GX sends the `GXOpenConnection` message to establish communications with a device in preparation for sending data to it. You can override the `GXOpenConnection` message to open a connection with a specific device. Your override of the `GXOpenConnection` message must match the following formal declaration:

```
OSErr GXOpenConnection (void);
```

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

**DESCRIPTION**

QuickDraw GX's default implementation of the `GXImageJob` message sends the `GXOpenConnection` message to prepare to send data to the device.

The default implementation of the `GXOpenConnection` message handles PAP, Serial, and not-connected connections. It reads the resource from the communications ('comm') resource in the desktop printer file.

## Printing Messages

You can override this message to open a connection with a specific device (such as a sheetfeeder) or to perform status checks at the time the device is opened. If you perform a status check and a failure occurs at that time, you must call the `GXCleanupOpenConnection` function.

**Note**

The default implementation of this message does not support SCSI devices. ♦

**SPECIAL CONSIDERATIONS**

You never send the `GXOpenConnection` message yourself.

If you are implementing a nonsupported type of communications connection, you need to perform a total override of the `GXOpenConnection` message. Otherwise, you need to first forward the message and then perform status checking on the device.

**RESULT CODES**

<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.

The default implementation of the `GXOpenConnection` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

**SEE ALSO**

The `GXImageJob` message is described on page 4-90.

You can find an example of an override of the `GXOpenConnection` message in Listing 3-8 on page 3-30 in the chapter “Printer Drivers.”

The `GXCleanupOpenConnection` function is described on page 5-36 in the chapter “Printing Functions for Message Overrides.”

The communications (`'comm'`) resource is described in the section “The Communications (`'comm'`) Resource” beginning on page 6-36 in the chapter “Printing Resources.”

**GXOpenConnectionRetry**

---

QuickDraw GX sends the `GXOpenConnectionRetry` message if opening the connection fails. You can override the `GXOpenConnectionRetry` message to try again to open the connection for a device. Your override of the `GXOpenConnectionRetry` message must match the following formal declaration:

Printing Messages

```
OSErr MyOpenConnectionRetry (ResType commType, void *commData,
                             Boolean *pRetry, OSErr saveErr);
```

- commType**     The resource type of the communications ('comm') resource to use. The possible values are shown in Table 4-9.
- commData**     A pointer to the communications resource that is stored with the desktop printer.
- pRetry**        On return, the Boolean value is true if QuickDraw GX needs to send the GXOpenConnection message again and false if not.
- saveErr**        The error code generated when the GXOpenConnection message failed.

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

DESCRIPTION

QuickDraw GX sends the GXOpenConnectionRetry message when a message handler calls the GXOpenConnection and it returns an error. You can override this message when you are implementing a communications protocol for a printer that can be shared by multiple users and accepts only one connection at a time. Your override can interpret the error that is specified in the saveErr parameter as an indication that the printer is in use by another user and that the connection attempt needs to be tried again.

The type of the communications resource, which you specify in the commType parameter, is one of the values shown in Table 4-9.

Table 4-9     Communications resource types

Constant	Value	Explanation
Serial	'SPTL'	Serial communications resource
PAP	'PPTL'	AppleTalk PAP communications resource
SCSI	'sPTL'	SCSI communications resource
PrinterShare	'ptsr'	PrinterShare communications resource
NotConnected	'Nops'	No communications resource

The default implementation of the OpenRetryConnection message sets the value of the pRetry return parameter to false for most devices. The exception is for PAP devices that are using the PostScript imaging system. In this case, the default implementation sets the pRetry parameter value to true if it determines that the printer is busy with another user's print job.

If your device automatically handles retrying connections, override this message and set the pRetry parameter to true.

**SPECIAL CONSIDERATIONS**

You never send the `GXOpenConnectionRetry` message yourself.

If you are implementing a nonsupported type of communications connection, you need to perform a total override of the `GXOpenConnectionRetry` message. Otherwise, you need to first forward the message and then perform your tasks.

**RESULT CODES**

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

**SEE ALSO**

The `GXOpenConnection` message is described in the previous section.

The communications ('comm') resource is described in the section “The Communications ('comm') Resource” beginning on page 6-36 in the chapter “Printing Resources.”

## **GXCleanupOpenConnection**

---

QuickDraw GX sends the `GXCleanupOpenConnection` message when an operation that has to be undone fails during the processing of an `GXOpenConnection` message. You need to override the `GXCleanupOpenConnection` message if you perform operations that must be undone after a failure in `GXOpenConnection`. Your override of the `GXCleanupOpenConnection` message must match the following formal declaration:

```
void MyCleanupOpenConnection (void);
```

**DESCRIPTION**

When an operation fails in your override of the `GXOpenConnection` message, you need to call the `GXCleanupOpenConnection` function.

When you call the `GXCleanupOpenConnection` function, QuickDraw GX sends the `GXCleanupOpenConnection` message to any message handlers that follow your printing extension or printer driver in the message chain. Each message handler is responsible for cleaning up any operations that it performed in its `GXOpenConnection` override. Usually this involves deallocating any storage that you allocated in your override of the `GXOpenConnection` message to open the connection.

The `GXCleanupOpenConnection` message follows the same path through the message chain as did the original `GXOpenConnection` message, which allows the cleaning up to occur in the correct order.



The default implementation of the `GXCleanupOpenConnection` message disposes of memory allocated by the default implementation of the `GXOpenConnection` message.

#### SPECIAL CONSIDERATIONS

You never send the `GXCleanupOpenConnection` message yourself; however, you can call the `GXCleanupOpenConnection` function, which then sends this message.

You need to forward the `GXCleanupOpenConnection` message so that other message handlers can perform their cleanup tasks.

#### SEE ALSO

The `GXCleanupOpenConnection` function is described on page 5-36 in the chapter “Printing Functions for Message Overrides” in this book.

The `GXOpenConnection` message is described in the previous section.

## GXCloseConnection

---

QuickDraw GX sends the `GXCloseConnection` message when all of the data for a print job has been sent to the device. You can override the `GXCloseConnection` message to perform actions your printing extension or printer driver requires at the completion of a job. Your override of the `GXCloseConnection` message must match the following formal declaration:

```
OSErr MyCloseConnection (void);
```

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

#### DESCRIPTION

The default implementation of the `GXImageJob` message sends the `GXCloseConnection` message at the end of sending to a device all of the data of a print job.

The default implementation of the `GXCloseConnection` message closes the connection to a printer. It handles PAP, serial, and not-connected connections. You can override this message to close a connection with a specific device, such as a sheetfeeder.

#### SPECIAL CONSIDERATIONS

You never send the `GXCloseConnection` message yourself.

If you are implementing a nonsupported type of communications connection, you need to perform a total override of the `GXCloseConnection` message. Otherwise, you need to perform your tasks and then forward the message.

## Printing Messages

## 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.

The default implementation of the `GXCloseConnection` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

## SEE ALSO

The `GXImageJob` message is described on page 4-90.

## GXStartSendPage

---

QuickDraw GX sends the `GXStartSendPage` message when it's ready to send a page to the printer. You can override the `GXStartSendPage` message to set up printing for the next page. Your override of the `GXStartSendPage` message must match the following formal declaration:

```
OSErr MyStartSendPage (gxFormat pageFormat);
```

`pageFormat` The format for this page.

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

## DESCRIPTION

QuickDraw GX's default implementation of the `GXImagePage` message sends the `GXStartSendPage` message to signal that QuickDraw GX is ready to start sending a page to the printer.

The default implementation of this message handles PAP, serial, and not-connected connections. You can override this message to perform tasks such as making sure that the previous page printed successfully and resetting page margins in the printer.

You must forward the `GXStartSendPage` message to other message handlers so that they can override it. If your override fails, you need to call the `GXCleanupStartSendPage` function to notify other handlers of the failure. If another handler returns an error, you must undo anything that you've done and return the same error.

## SPECIAL CONSIDERATIONS

You never send the `GXStartSendPage` message yourself.

If you are implementing a nonsupported type of communications connection, you need to perform a total override of the `GXStartSendPage` message. Otherwise, you need to first forward the message and then perform your tasks.

**RESULT CODES**

<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.

The default implementation of the `GXStartSendPage` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

**SEE ALSO**

You can find an example of an override of the `GXStartSendPage` message in Listing 3-13 on page 3-39 in the chapter “Printer Drivers.”

The `GXCleanupStartSendPage` function is described on page 5-37 in the chapter “Printing Functions for Message Overrides.”

The `GXImagePage` message is described on page 4-94.

## **GXCleanupStartSendPage**

---

QuickDraw GX sends the `GXCleanupStartSendPage` message when an operation that has to be undone fails during the processing of a `GXStartSendPage` message. You need to override the `GXCleanupStartSendPage` message if you have overridden the `GXStartSendPage` message and your override contains code that can fail, such as a memory allocation or device initialization. Your override of the `GXCleanupStartSendPage` message must match the following formal declaration:

```
void MyCleanupStartSendPage (void);
```

**DESCRIPTION**

When an operation fails in your override of the `GXStartSendPage` message, you need to call the `GXCleanupStartSendPage` function.

When you call the `GXCleanupStartSendPage` function, QuickDraw GX sends the `GXCleanupStartSendPage` message to any message handlers that follow your printing extension or printer driver in the message chain. Each message handler is responsible for cleaning up any operations that it performed in its `GXStartSendPage` override. Usually this involves deallocating any storage that you allocated in your override of the `GXStartSendPage` message.

## Printing Messages

The `GXCleanupStartSendPage` message follows the same path through the message chain as did the original `GXStartSendPage` message, which allows the cleaning up to occur in the correct order.

The default implementation of the `GXCleanupStartSendPage` message disposes of memory allocated by the default implementation of the `GXStartSendPage` message.

**SPECIAL CONSIDERATIONS**

You never send the `GXCleanupStartSendPage` message yourself; however, you can call the `GXCleanupStartSendPage` function, which then sends this message.

You must forward the `GXCleanupStartSendPage` message to allow other message handlers to clean up any storage that they allocated in response to the `GXStartSendPage` message.

**SEE ALSO**

The `GXCleanupStartSendPage` function is described on page 5-37 in the chapter “Printing Functions for Message Overrides.”

The `GXStartSendPage` message is described in the previous section.

**GXFinishSendPage**

---

QuickDraw GX sends the `GXFinishSendPage` message after all of the data for a page has been sent to the printer. You can override the `GXFinishSendPage` message to perform some action at the end of a page. Your override of the `GXFinishSendPage` message must match the following formal declaration:

```
OSErr MyFinishSendPage (void);
```

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

**DESCRIPTION**

QuickDraw GX’s default implementation of the `GXImagePage` message sends the `GXFinishSendPage` message after the page is sent to the printer and before it starts sending the next page.

The default implementation of the `GXFinishSendPage` message handles PAP, serial, and not-connected connections. You override this message to perform some action at the end of the page, such as telling the printer to print the page that is loaded into its memory or prompting the user to add paper for a manual-feed print job.

**SPECIAL CONSIDERATIONS**

You never send the `GXFinishSendPage` message yourself.

If you are implementing a nonsupported type of communications connection, you need to perform a total override of the `GXFinishSendPage` message. Otherwise, you need to perform your tasks and then forward the message.

**RESULT CODES**

<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.

The default implementation of the `GXFinishSendPage` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

**SEE ALSO**

The `GXImagePage` message is described on page 4-94.

**GXBufferData**

QuickDraw GX sends the `GXBufferData` message to send data to the device using the standard buffering mechanism. You can override the `GXBufferData` message to provide your own buffering scheme or to change data prior to buffering. Your override of the `GXBufferData` message must match the following formal declaration:

```
OSErr MyBufferData (Ptr data, long length, long bufferOptions);
```

`data`            A pointer to the data to add to the buffer.

`length`        The number of bytes of data to add.

`bufferOptions`       Options for adding the data to the buffer, as shown in Table 4-10.

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

**DESCRIPTION**

QuickDraw GX sends the `GXBufferData` message in preparation for sending data to a device using the standard buffering mechanism. This message is sent whenever any message handler wishes to buffer the data to be sent to the printing device.

## Printing Messages

The default implementation of this message uses the default buffering. It buffers the data until all of the buffers are full. You can specify buffering options in the `bufferOptions` parameter. The constants for these options are shown in Table 4-10.

**Table 4-10** Options for adding data to the buffer

Constant	Value	Explanation
<code>gxNoBufferOptions</code>	<code>0x00000000</code>	No options apply.
<code>gxMakeBufferHex</code>	<code>0x00000001</code>	All data is converted into ASCII hex equivalents: each byte becomes 2 ASCII bytes.
<code>gxDontSplitBuffer</code>	<code>0x00000002</code>	The data sent in this message cannot be split across buffers. If it is too large to fit into what is left of the current buffer, then add it to the next buffer instead.

When the buffers are completely full, QuickDraw GX sends the `GXDumpBuffer` message to write the data in the buffer to the device and then sends the `GXFreeBuffer` message to open the buffer to write additional data.

## SPECIAL CONSIDERATIONS

You can send the `GXBufferData` message to add data to the printer's data stream.

If you are implementing your own buffering scheme, you need to perform a total override of the `GXBufferData` message. Otherwise, you need to modify the data and then forward the message.

## RESULT CODES

<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.
<code>gxGBBufferTooSmallErr</code>	The buffer is too small to accept the requested data, and you have specified that buffers are not to be split.

The default implementation of the `GXBufferData` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

## SEE ALSO

The `GXDumpBuffer` message is described on page 4-142.

The `GXFreeBuffer` message is described on page 4-143.

## GXWriteData

---

QuickDraw GX sends the `GXWriteData` message to send data directly to the device, without buffering. You can override the `GXWriteData` message to provide your own I/O mechanism. Your override of the `GXWriteData` message must match the following formal declaration:

```
OSErr MyWriteData (Ptr data, long length);
```

`data`            A pointer to the data to send to device.

`length`        The number of bytes of data.

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

### DESCRIPTION

The `GXWriteData` message sends unbuffered data to a device and waits for I/O to complete.

The default implementation of this message calls the asynchronous I/O functions of the Macintosh system software to send the data to the device. The default implementation then flushes the buffers, sends the resulting data through, and waits for the I/O operation to complete.

You can override this message as part of implementing your own I/O and buffering mechanism or changing how I/O is performed with your device.

### SPECIAL CONSIDERATIONS

You can use the `GXWriteData` message yourself to send data to your device without buffering or to force the buffers to be flushed.

If you are implementing your own I/O mechanism, you need to totally override the `GXWriteData` message. If you are using the standard QuickDraw GX buffering and Macintosh I/O system, you must forward this message.

### RESULT CODES

<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.

The default implementation of the `GXWriteData` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

## GXDumpBuffer

---

QuickDraw GX sends the `GXDumpBuffer` message in order to write to the device the data in the buffers. You can override the `GXDumpBuffer` message to process data and buffers in your own way. Your override of the `GXDumpBuffer` message must match the following formal declaration:

```
OSErr MyDumpBuffer (gxPrintingBuffer *aPrintingBuffer);
```

`aPrintingBuffer`

A pointer to the printing buffer structure that defines the buffer to send to the device.

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

### DESCRIPTION

QuickDraw GX sends the `GXDumpBuffer` message to write data to the device. It sends the message when any one of the buffers is full and data must be sent.

The default implementation of this message calls the asynchronous I/O functions of the Macintosh system software to send the buffer to the device. The default implementation then sends the requested buffer to the device using PAP, serial, or not-connected communications, depending on what you specified in your printing extension or printer driver resources. You can override this message to process data and buffers in your own way.

### SPECIAL CONSIDERATIONS

You can send the `GXDumpBuffer` message yourself if you are implementing your own buffering scheme.

If you are implementing your own I/O mechanism, you need to perform a total override of the `GXDumpBuffer` message. Otherwise, you need to forward this message to the other printing message handlers.

If you override the `GXDumpBuffer` message, you must also override the `GXFreeBuffer` message.

### RESULT CODES

<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.

The default implementation of the `GXDumpBuffer` message can also return the communications errors that are listed in Table 4-2 on page 4-42.



**SEE ALSO**

The `GXFreeBuffer` message is described in the next section.

The printing buffer structure is described on page 4-11.

**GXFreeBuffer**

---

QuickDraw GX sends the `GXFreeBuffer` message when waiting for the completion of a buffer that has been sent with the `GXDumpBuffer` message. You can override the `GXFreeBuffer` message if you are processing data and buffers in your own way. Your override of the `GXFreeBuffer` message must match the following formal declaration:

```
OSErr MyFreeBuffer (gxPrintingBufferPtr *aPrintingBuffer);
```

`aPrintingBuffer`

A pointer to the printing buffer structure that defines the buffer to wait for.

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

**DESCRIPTION**

The default implementation of the `GXFreeBuffer` message waits for processing of the buffer to complete using PAP, serial, or not-connected communications. It calls the asynchronous I/O functions of the Macintosh system software to wait for the buffer to complete transmission to the device. QuickDraw GX sends this message when it needs a buffer and they are all full.

You can override this message if you are processing data and buffers in your own way. You must not return from this message until the I/O completes or terminates with an error.

**SPECIAL CONSIDERATIONS**

You can send the `GXFreeBuffer` message yourself if you are implementing your own buffering scheme.

If you override the `GXDumpBuffer` message, you must also override the `GXFreeBuffer` message.

If you are implementing your own I/O mechanism, you need to perform a total override of the `GXFreeBuffer` message.

## Printing Messages

## 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.

The default implementation of the `GXFreeBuffer` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

## SEE ALSO

The `GXDumpBuffer` message is described in the previous section.

The printing buffer structure is described on page 4-11.

## GXFinishSendPlane

---

QuickDraw GX sends the `GXFinishSendPlane` message to mark the end of processing of a plane or “color pass” for raster devices. You can override the `GXFinishSendPlane` message to finish processing of a color plane and send it to your device. Your override of the `GXFinishSendPlane` message must match the following formal declaration:

```
OSErr MyFinishSendPlane (void);
```

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

## DESCRIPTION

QuickDraw GX sends this message to mark the end of a color plane for raster devices that require multiple passes for each page. A film recorder might send this message to mark the end of the red, green, and blue color passes.

QuickDraw GX does not have a default implementation of the `GXFinishSendPlane` message.

## RESULT CODES

<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.

## SPECIAL CONSIDERATIONS

You can send the `GXFinishSendPlane` message yourself to mark completion of a plane.

You can totally override the `GXFinishSendPlane` message to perform tasks at the completion of a color plane pass.

## GXCheckStatus

---

QuickDraw GX sends the `GXCheckStatus` message to mark a location in the communications process where a status check needs to be performed. You need to override the `GXCheckStatus` message to check the status of your device. Your override of the `GXCheckStatus` message must match the following formal declaration:

```
OSErr MyCheckStatus (Ptr data, long length, short statusType,
                    Signature owner);
```

<code>data</code>	A pointer to the data that you want to send with the status.
<code>length</code>	The number of bytes in the data.
<code>statusType</code>	The application-defined status type.
<code>owner</code>	The ID of the message handler for which the <code>GXCheckStatus</code> message is intended.

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

### DESCRIPTION

You can send the `GXCheckStatus` message to mark the point in the imaging-communications process where you can check status information about the device.

The default implementation of this message does nothing. You need to override this message to provide status checking for your device.

The message handler that sends the `GXCheckStatus` message sets the `data`, `length`, and `statusType` parameters. You need to check the `owner` parameter and only process the status check if it matches your signature; if not, you need to forward the message. Most message handlers call the `GXGetDeviceStatus` message in response to the `GXCheckStatus` message.

### SPECIAL CONSIDERATIONS

You can send the `GXCheckStatus` message yourself to mark a place where you want to have a status check performed.

If you process the status check in your override, you do not forward the `GXCheckStatus` message. If you do not handle the status check, you must forward the message.

## Printing Messages

## RESULT CODES

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

## SEE ALSO

The `GXGetDeviceStatus` message is described in the next section.

## GXGetDeviceStatus

---

QuickDraw GX sends the `GXGetDeviceStatus` message to query the device about its status. You can override the `GXGetDeviceStatus` message to send a query to your own device about its status. Your override of the `GXGetDeviceStatus` message must match the following formal declaration:

```
OSErr MyGetDeviceStatus (Ptr cmdData, long cmdSize,
                        Ptr responseData, long *responseSize,
                        Str255 termination);
```

`cmdData`      A pointer to the status query command data.

`cmdSize`      The number of bytes in the command data.

`responseData`  
On return, a pointer to the data that contains the device's response.

`responseSize`  
On entry, the number of bytes to read. On return, the number of bytes in the response data.

`termination`  
This is the string that marks the end of the response in the `responseData` parameter. If this is `nil`, you need to read all of the bytes in the response data.

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

## DESCRIPTION

Before sending the `GXGetDeviceStatus` message yourself, you need to make sure that all pending I/O with your device is flushed. This can be done by sending a `GXWriteData` message with a `nil` pointer and a length of 0.

The default implementation of this message supports PAP, serial, and not-connected communications by writing a status request and reading back the result from the device.

**SPECIAL CONSIDERATIONS**

You can send the `GXGetDeviceStatus` message when you want to perform a status check of a device.

If you are implementing a nonsupported type of communications connection, you need to perform a total override of the `GXGetDeviceStatus` message. Otherwise, you need to perform your tasks and then forward the message.

**RESULT CODES**

<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.

The default implementation of the `GXGetDeviceStatus` message can also return the communications errors that are listed in Table 4-2 on page 4-42.

**SEE ALSO**

The `GXWriteData` message is described on page 4-141.

**Compatibility Messages**

QuickDraw GX sends these messages only when an application makes calls to the Macintosh Printing Manager. QuickDraw GX responds to these messages by mapping the Macintosh Printing Manager behavior into QuickDraw GX actions. If you are writing drivers, these message allow you to customize the behavior of these printing calls, although you can most easily accomplish the basic mapping with the resources that are described in the chapter “Printing Resources” in this book.

If you need to learn about the Macintosh Printing Manager and how these calls were used to implement QuickDraw printing, refer to *Inside Macintosh: Imaging With QuickDraw*.

QuickDraw GX looks for dialog resources ('DLOG', 'DITL', and 'dctl' resources) with specific resource IDs to find the dialog information that is used by some of the compatibility messages, including the `GXPrStlDialog`, `GXPrJobDialog`, `GXPrStlInit`, `GXPrJobInit`, and `GXPrDlgMain` functions. QuickDraw GX looks for dialog resources with ID = -8192 for style dialog information and with ID = -8191 for job dialog information. If QuickDraw GX does not find dialog resources with these IDs in the driver resources, it uses default versions. The dialog control ('dctl') resource is described in the section “The Dialog Control ('dctl') Resource” beginning on page 6-52 in the chapter “Printing Resources.” The dialog ('DLOG') and dialog item list ('DITL') resources are described in *Inside Macintosh: Macintosh Toolbox Essentials*.

## GXPrOpenDoc

---

QuickDraw GX sends the `GXPrOpenDoc` message when an application that supports the Macintosh Printing Manager calls the `PrOpenDoc` function. You can override the `GXPrOpenDoc` message to customize the handling of the `PrOpenDoc` function. Your override of the `GXPrOpenDoc` message must match the following formal declaration:

```
OSErr MyPrOpenDoc (THPrint aTHPrint, TPrPort *aTPrPort);
```

`aTHPrint`     A handle to the print record for this printing operation.

`aTPrPort`     A pointer to a `TPrPort` record.

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

### DESCRIPTION

The default implementation of the `GXPrOpenDoc` sends the `GXStartJob` message. You can override the `GXPrOpenDoc` message to add any special handling that your printer driver requires at the time that a document is spooled for printing.

### SPECIAL CONSIDERATIONS

You never send the `GXPrOpenDoc` message yourself.

You almost always forward the `GXPrOpenDoc` message so that the default implementation can perform its operations.

### RESULT CODES

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

### SEE ALSO

The `PrOpenDoc` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The `GXStartJob` message is described on page 4-52.

## GXPrCloseDoc

---

QuickDraw GX sends the `GXPrCloseDoc` message when an application that supports the Macintosh Printing Manager calls the `PrCloseDoc` function. You can override the

## Printing Messages

GXPrCloseDoc message to customize the handling of the PrCloseDoc function. Your override of the GXPrCloseDoc message must match the following formal declaration:

```
OSErr MyPrCloseDoc (TPPrPort aTPPrPort);
```

aTPPrPort    A pointer to a TPPrPort record.

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

**DESCRIPTION**

The default implementation of the GXPrCloseDoc message generates the GXFinishJob message. You can override this message to add any special handling that your printer driver requires at the time that a document is finished printing.

**SPECIAL CONSIDERATIONS**

You never send the GXPrCloseDoc message yourself.

You almost always forward the GXPrCloseDoc message so that the default implementation can perform its operations.

**RESULT CODES**

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

**SEE ALSO**

The PrCloseDoc function is described in *Inside Macintosh: Imaging With QuickDraw*.

The GXFinishJob message is described on page 4-54.

**GXPrOpenPage**

QuickDraw GX sends the GXPrOpenPage message when an application that supports the Macintosh Printing Manager calls the PrOpenPage function. You can override the GXPrOpenPage message to customize the handling of the PrOpenPage function. Your override of the GXPrOpenPage message must match the following formal declaration:

```
OSErr MyPrOpenPage (TPPrPort aTPPrPort, TPRect aTPRect,
                    Point resolution);
```

aTPPrPort    A pointer to a TPPrPort record.

## Printing Messages

`aTPRect` The rectangle used as the QuickDraw picture frame for this page.

`resolution` The resolution to be used in printing the page.

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

## DESCRIPTION

The default implementation of the `GXPrOpenPage` message generates the `GXStartPage` message. You can override this message to add any special handling that your printer driver requires at the time that a page is spooled for printing.

## SPECIAL CONSIDERATIONS

You never send the `GXPrOpenPage` message yourself.

You almost always forward the `GXPrOpenPage` message so that the default implementation can perform its operations.

## RESULT CODES

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

## SEE ALSO

The `PrOpenPage` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The `GXStartPage` message is described on page 4-55.

## GXPrClosePage

---

QuickDraw GX sends the `GXPrClosePage` message when an application that supports the Macintosh Printing Manager calls the `PrClosePage` function. You can override the `GXPrClosePage` message to customize the handling of the `PrClosePage` function. Your override of the `GXPrClosePage` message must match the following formal declaration:

```
OSErr GXPrCloseDoc (TPPrPort aTPPrPort);
```

`aTPPrPort` A pointer to a `TPPrPort` record.

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



**DESCRIPTION**

The default implementation of this message generates the `GXFinishPage` message. You can override this message to add any special handling that your driver requires at the time that a page has finished printing.

**SPECIAL CONSIDERATIONS**

You never send the `GXPrClosePage` message yourself.

You almost always forward the `GXPrClosePage` message so that the default implementation can perform its operations.

**RESULT CODES**

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

**SEE ALSO**

The `PrClosePage` function is described in *Inside Macintosh: Imaging With QuickDraw*.  
The `GXFinishPage` message is described on page 4-57.

**GXPrintDefault**

---

QuickDraw GX sends the `GXPrintDefault` message when an application that supports the Macintosh Printing Manager calls the `PrintDefault` function. You can override the `GXPrintDefault` message to customize the handling of the `PrintDefault` function. Your override of the `GXPrintDefault` message must match the following formal declaration:

```
OSErr MyPrintDefault (THPrint aTHPrint);
```

`aTHPrint`     A handle to the print record for this printing operation.

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

**DESCRIPTION**

The default implementation of this message loads a default 'PREC' resource. You can override this message to add any special handling of the default print record.

**SPECIAL CONSIDERATIONS**

You never send the `GXPrintDefault` message yourself.

You almost always forward the `GXPrintDefault` message so that the default implementation can perform its operations.

**RESULT CODES**

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

**SEE ALSO**

The `PrintDefault` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The 'PREC' resource is described on page 6-51 in the chapter “Printing Resources” in this book.

**GXPrStlDialog**

---

QuickDraw GX sends the `GXPrStlDialog` message when an application that supports the Macintosh Printing Manager calls the `PrStlDialog` function. You can override the `GXPrStlDialog` message to customize the handling of the `PrStlDialog` function. Your override of the `GXPrStlDialog` message must match the following formal declaration:

```
OSErr MyPrStlDialog (THPrint aTHPrint, Boolean *aBoolean);
```

`aTHPrint`     A handle to the print record for this printing operation.

`aBoolean`     On return, a Boolean value that is `true` if the user confirmed the dialog box and `false` if not.

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

**DESCRIPTION**

The default implementation of the `GXPrStlDialog` message looks for dialog ('DLOG'), item list ('DITL'), and dialog control ('dctl') resources that have the ID -8192 in your driver resources. If QuickDraw GX does not find resources with this ID, it uses its own default resources to construct and display a style dialog box. Some of the values displayed in the style dialog box are controlled by the customization ('cust') resource.

You can override the `GXPrStlDialog` message to add any special handling that your printer driver requires with regard to page dimensions and page setup.

**SPECIAL CONSIDERATIONS**

You never send the `GXPrStlDialog` message yourself.

You almost always forward the `GXPrStlDialog` message so that the default implementation can perform its operations.

**RESULT CODES**

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

**SEE ALSO**

The `PrStlDialog` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The dialog control and customization resources are described in the chapter “Printing Resources” in this book.

The dialog and item list resources are described in *Inside Macintosh: Macintosh Toolbox Essentials*.

**GXPrJobDialog**

QuickDraw GX sends the `GXPrJobDialog` message when an application that supports the Macintosh Printing Manager calls the `PrJobDialog` function. You can override the `GXPrJobDialog` message to customize the handling of the `PrJobDialog` function. Your override of the `GXPrJobDialog` message must match the following formal declaration:

```
OSErr MyPrJobDialog (THPrint aTHPrint, Boolean *aBoolean);
```

`aTHPrint`     A handle to the print record for this printing operation.

`aBoolean`     On return, a Boolean value that is `true` if the user confirmed the dialog box and `false` if not.

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

**DESCRIPTION**

The default implementation of the `GXPrJobDialog` message looks for dialog (`'DLOG'`), item list (`'DITL'`), and dialog control (`'dctl'`) resources that have the ID -8191 in your driver resources. If QuickDraw GX does not find resources with this ID, it uses its own default resources to construct and display a Print dialog box. Some of the values displayed in the Print dialog box are controlled by the customization (`'cust'`) resource.

## Printing Messages

You can override this message to add any special handling that your driver requires with regard to print quality, range of pages, and other properties associated with a print object.

## SPECIAL CONSIDERATIONS

You never send the `GXPrJobDialog` message yourself.

You almost always forward the `GXPrJobDialog` message so that the default implementation can perform its operations.

## RESULT CODES

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

## SEE ALSO

The `PrJobDialog` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The dialog control and customization resources are described in the chapter “Printing Resources” in this book.

The dialog and item list resources are described in *Inside Macintosh: Macintosh Toolbox Essentials*.

## GXPrStlInit

---

QuickDraw GX sends the `GXPrStlInit` message when an application that supports the Macintosh Printing Manager calls the `PrStlInit` function to add controls to the style dialog box that is displayed when the `PrDlgMain` function is called. You can override the `GXPrStlInit` message to customize the handling of the `PrStlInit` function. Your override of the `GXPrStlInit` message must match the following formal declaration:

```
OSErr MyPrStlInit (THPrint aTHPrint, TPPrDlg *aTPPrDlg);
```

`aTHPrint`     A handle to the print record for this printing operation.

`aTPPrDlg`     A pointer to the dialog structure for the style dialog box.

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

## DESCRIPTION

The default implementation of the `GXPrStlInit` message sets up default controls in the style dialog box. You can override this message if you need to add any special handling to the processing of `PrStlInit` calls by your printer driver.

**SPECIAL CONSIDERATIONS**

You never send the `GXPrStlInit` message yourself.

You almost always forward the `GXPrStlInit` message so that the default implementation can perform its operations.

**RESULT CODES**

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

**SEE ALSO**

The `PrStlInit` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The dialog control and customization resources are described in the chapter “Printing Resources” in this book.

The dialog and item list resources are described in *Inside Macintosh: Macintosh Toolbox Essentials*.

**GXPrJobInit**

---

QuickDraw GX sends the `GXPrJobInit` message when an application that supports the Macintosh Printing Manager calls the `PrJobInit` function to set up the structure that is used for displaying the Print dialog box when the `PrDlgMain` function is called. You can override the `GXPrJobInit` message to customize the handling of the `GXPrJobInit` function. Your override of the `GXPrJobInit` message must match the following formal declaration:

```
OSErr MyPrJobInit (THPrint aTHPrint, TPPrDlg *aTPPrDlg);
```

`aTHPrint`     A handle to the print record for this printing operation.

`aTPPrDlg`     A pointer to the `TPPrDlg` record for the Print dialog box.

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

**DESCRIPTION**

The default implementation of the `GXPrJobInit` message sets up default controls in the Print dialog box. You can override this message if you need to add any special handling to the processing of `PrJobInit` calls by your printer driver.

## Printing Messages

## SPECIAL CONSIDERATIONS

You never send the `GXPrJobInit` message yourself.

You almost always forward the `GXPrJobInit` message so that the default implementation can perform its operations.

## RESULT CODES

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

## SEE ALSO

The `PrJobInit` function is described in *Inside Macintosh: Imaging With QuickDraw*.

## GXPrDlgMain

---

QuickDraw GX sends the `GXPrDlgMain` message when an application that supports the Macintosh Printing Manager calls the `PrDlgMain` function. You can override the `GXPrDlgMain` message to customize the handling of the `PrDlgMain` function. Your override of the `GXPrDlgMain` message must match the following formal declaration:

```
OSErr MyPrDlgMain (THPrint aTHPrint,
                  PDlgInitProcPtr aPDlgInitProcPtr,
                  Boolean *aBoolean);
```

`aTHPrint`     A handle to the print record for this printing operation.

`aPDlgInitProcPtr`     A pointer to the procedure used to initialize the Print dialog box.

`aBoolean`     On return, a Boolean value that is `true` if the user confirmed the dialog box and `false` if not.

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

## DESCRIPTION

The default implementation of the `GXPrDlgMain` message sets up the default controls for the Print dialog box. You can override this message if you need to add any special handling to the processing of `GXPrDlgMain` calls by your printer driver.

## SPECIAL CONSIDERATIONS

You never send the `GXPrDlgMain` message yourself.

## Printing Messages

You almost always forward the `GXPrDlgMain` message so that the default implementation can perform its operations.

## RESULT CODES

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

## SEE ALSO

The `PrDlgMain` function is described in *Inside Macintosh: Imaging With QuickDraw*.

## GXPrValidate

---

QuickDraw GX sends the `GXPrValidate` message when an application that supports the Macintosh Printing Manager calls the `PrValidate` function. You can override the `GXPrValidate` message to customize the handling of the `PrValidate` function. Your override of the `GXPrValidate` message must match the following formal declaration:

```
OSErr MyPrValidate (THPrint aTHPrint, Boolean *aBoolean);
```

<code>aTHPrint</code>	A handle to the print record for this printing operation.
<code>aBoolean</code>	On return, a Boolean value that is <code>true</code> if the user confirmed the dialog box and <code>false</code> if not.

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

**IMPORTANT**

The default implementation of this message sends a `GXPrintDefault` message if the value of the `devKind` field in the print record is other than `0xA900` or if the value of the `prVersion` field in the print record is other than 8. ▲

## DESCRIPTION

The default implementation of this message validates the print record using the universal print structure. It sends the `GXConvertPrintRecordTo` message, validates the contents of the structure, and then sends the `GXConvertPrintRecordFrom` message.

You can override this message if you need to add any special handling to the processing of `PrValidate` calls by your printer driver.

**SPECIAL CONSIDERATIONS**

You never send the `GXPrValidate` message yourself.

You only forward the `GXPrValidate` message so that the default implementation can perform its operations if you do not have a custom print-record format.

The customization resource is described on page 6-47 in the chapter “Printing Resources” in this book.

**RESULT CODES**

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

**SEE ALSO**

The `PrValidate` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The universal print structure is described in the section “The Universal Print Structure” on page 4-12.

The `GXConvertPrintRecordTo` message is described page 4-161.

The `GXConvertPrintRecordFrom` message is described on page 4-160.

**GXPrGeneral**

---

QuickDraw GX sends the `GXPrGeneral` message when an application that supports the Macintosh Printing Manager calls the `PrGeneral` function. You can override the `GXPrGeneral` message to customize the handling of the `PrGeneral` function. Your override of the `GXPrGeneral` message must match the following formal declaration:

```
OSErr MyPrGeneral (Ptr aPtr);
```

`aPtr`            A pointer to the data block used by `GXPrGeneral`.

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

**DESCRIPTION**

The default implementation of this message uses the customization (`'cust'`) resource to find the supported resolution for the printer. It also uses any available resolution (`'resl'`) resources. You can override this message if you need to add any special handling to the processing of `GXPrGeneral` calls by your printer driver.



**SPECIAL CONSIDERATIONS**

You never send the `GXPrGeneral` message yourself.

You almost always forward the `GXPrGeneral` message so that the default implementation can perform its operations.

**RESULT CODES**

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

**SEE ALSO**

The `PrGeneral` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The customization and resolution resources are described in the chapter “Printing Resources” in this book.

**GXPrJobMerge**

---

QuickDraw GX sends the `GXPrJobMerge` message when an application that supports the Macintosh Printing Manager calls the `PrJobMerge` function. You can override the `GXPrJobMerge` message to customize the handling of the `PrJobMerge` function. Your override of the `GXPrJobMerge` message must match the following formal declaration:

```
OSErr MyPrJobMerge (THPrint aTHPrint1, THPrint aTHPrint2);
```

`aTHPrint1`    A handle to the first print record for this printing operation.

`aTHPrint2`    A handle to the second print record for this printing operation.

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

**DESCRIPTION**

The default implementation of this message merges the two print records and calls `GXConvertPrintRecordTo` to add the print information to the job collection so that you can print several documents with a single dialog box. You can override this message if you need to add any special handling to the processing of `GXPrJobMerge` calls for your driver.

**SPECIAL CONSIDERATIONS**

You never send the `GXPrJobMerge` message yourself.

## Printing Messages

You almost always forward the `GXPrJobMerge` message so that the default implementation can perform its operations.

## RESULT CODES

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

## SEE ALSO

The `PrJobMerge` function is described in *Inside Macintosh: Imaging With QuickDraw*.

The `GXConvertPrintRecordTo` function is described in *Inside Macintosh: QuickDraw GX Printing*.

## GXConvertPrintRecordFrom

---

QuickDraw GX sends the `GXConvertPrintRecordFrom` message when an application that supports the Macintosh Printing Manager calls the `GXConvertPrintRecordFrom` function, which converts a print record into a driver-specific format. You can override the `GXConvertPrintRecordFrom` message to customize the handling of the `GXConvertPrintRecordFrom` function. Your override of the `GXConvertPrintRecordFrom` message must match the following formal declaration:

```
OSErr MyConvertPrintRecordFrom (THPrint aTHPrint);
```

`aTHPrint`     A handle to the print record for this printing operation.

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

## DESCRIPTION

The default implementation of the `GXConvertPrintRecordFrom` message does nothing. Your override fills in the print record with information from the job collection.

**SPECIAL CONSIDERATIONS**

You never send the `GXConvertPrintRecordFrom` message yourself.

You need to totally override the `GXConvertPrintRecordFrom` message to support your Macintosh Printing Manager print-record format.

**RESULT CODES**

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

**SEE ALSO**

The `GXConvertPrintRecordFrom` function is described in *Inside Macintosh: QuickDraw GX Printing*.

**GXConvertPrintRecordTo**

---

QuickDraw GX sends the `GXConvertPrintRecordTo` message when an application that supports the Macintosh Printing Manager calls the `GXConvertPrintRecordTo` function, which converts a print record into a universal print structure format. You can override the `GXConvertPrintRecordTo` message if you have a printer driver written for the Macintosh Printing Manager whose print format you wish to continue to support. Your override of the `GXConvertPrintRecordTo` message must match the following formal declaration:

```
OSErr MyConvertPrintRecordTo (THPrint aTHPrint);
```

`aTHPrint`     A handle to the print record for this printing operation.

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

**DESCRIPTION**

The default implementation of the `GXConvertPrintRecordTo` message does nothing.

**SPECIAL CONSIDERATIONS**

You never send the `GXConvertPrintRecordTo` message yourself.

You need to totally override the `GXConvertPrintRecordTo` message to support your Macintosh Printing Manager print-record format.

## Printing Messages

## RESULT CODES

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

## SEE ALSO

The `GXConvertPrintRecordFrom` function is described in *Inside Macintosh: QuickDraw GX Printing*.

The universal print structure is described on page 4-12.

**GXPrintRecordToJob**

---

QuickDraw GX sends the `GXPrintRecordToJob` message when an application that supports the Macintosh Printing Manager calls the `GXPrintRecordToJob` function, which converts a print record into information in the job collection. You can override the `GXPrintRecordToJob` message if you have a printer driver written for the Macintosh Printing Manager with a print record format you wish to continue to support. Your override of the `GXPrintRecordToJob` message must match the following formal declaration:

```
OSErr MyPrintRecordToJob (THPrint aTHPrint, gxJob aJob);
```

`aTHPrint`     A handle to the print record for this printing operation.

`aJob`         The job object.

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

## DESCRIPTION

The default implementation of the `GXPrintRecordToJob` message sends the `GXConvertPrintRecordTo` message and then moves all of the universal fields into the job object.

## SPECIAL CONSIDERATIONS

You never send the `GXPrintRecordToJob` message yourself.

You can totally or partially override the `GXPrintRecordToJob` message.

## RESULT CODES

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

## SEE ALSO

The `GXPrintRecordToJob` function is described in *Inside Macintosh: QuickDraw GX Printing*.

The `GXConvertPrintRecordTo` message is described in the previous section.

## Finder Dialog Box Messages

---

The Finder uses Finder dialog box messages when it wants to display information in a dialog box for the user when there are messages or a printing problem in the background. It sends these messages during the Finder interaction phase of printing.

## GXWriteStatusToDTPWindow

---

The Finder sends the `GXWriteStatusToDTPWindow` message when it receives the status from a background printing process and wants to display that status in the user's desktop printer window. You can override the `GXWriteStatusToDTPWindow` message to determine if you want to handle a specific status message. Your override of the `WriteStatusToDTPWindow` message must match the following formal declaration:

```
OSErr MyWriteStatusToDTPWindow (gxStatusRecord aStatusRecord,
                                gxDisplayRecord aDisplayRecord);
```

`aStatusRecord`

A pointer to the status structure.

`aDisplayRecord`

On return, a pointer to the information to be displayed.

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

## DESCRIPTION

The default implementation of this message turns the status structure into a string using the status (`'stat'`) resource supplied by the driver.

Your override of the `GXWriteStatusToDTPWindow` message needs to examine the `statusOwner` field in the status structure to determine if you need to handle the status. If your override needs to display status information that is more complex than is supported by this standard status handling, you must fill in this structure yourself. If the `statusOwner` field in the status structure does not match your driver's signature, you must forward this message.

**SPECIAL CONSIDERATIONS**

You never send the `GXWriteStatusToDTPWindow` message yourself.

If you do handle the status, you need to perform a total override of the `GXWriteStatusToDTPWindow` message. Otherwise, you must forward the `GXWriteStatusToDTPWindow` message so that another handler can respond to it.

**RESULT CODES**

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

**SEE ALSO**

The status structure is described on page 4-39, and the status resource is described on page 6-19 in the chapter “Printing Resources.”

The display structure is described on page 4-41.

**GXInitializeStatusAlert**

---

QuickDraw GX sends the `GXInitializeStatusAlert` message when a driver calls the `GXAlertTheUser` function with a status structure containing a status type of `gxUserAlert`. You need to override the `GXInitializeStatusAlert` message if you want to provide custom alerts. Your override of the `GXInitializeStatusAlert` message needs to be declared as follows.

```
OSErr MyInitializeStatusAlert (gxStatusRecord *statRecPtr,
                             DialogPtr *dPtr);
```

`statRecPtr`           A pointer to the status structure.

`dPtr`                A pointer to the dialog box pointer.

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

**DESCRIPTION**

The default implementation of the `GXInitializeStatusAlert` message is the automatic handling of the standard printing ('plrt') alert boxes.

You do not need to override this message if your alert is one of the standard alerts for which QuickDraw GX provides automatic handling, as described in the chapter “Printer Drivers” in this book. If you want to provide custom alerts, first check the

## Printing Messages

`statusOwner` field of the status structure to make sure that this message is intended for you. If so, perform a total override of this message. You must create a dialog box and return a pointer to it. You can call the `GetNewDialog` function for the specified alert and then perform any necessary dialog box initialization.

## SPECIAL CONSIDERATIONS

You never send the `GXInitializeStatusAlert` message yourself.

If the `statusOwner` field of the status structure matches your signature, you perform a total override of the `GXInitializeStatusAlert` message. Otherwise, forward the message so that another message handler can respond to it.

## RESULT CODES

<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

You can find an example of an override of the `GXInitializeStatusAlert` message in Listing 3-17 on page 3-45 in the chapter “Printer Drivers.”

The `GXAlertTheUser` function is described on page 5-18 in the chapter “Printing Functions for Message Overrides.”

The `GetNewDialog` function is described in *Inside Macintosh: Macintosh Toolbox Essentials*.

The status structure is described on page 4-39.

## GXHandleAlertEvent

You need to override the `GXHandleAlertEvent` message if you are providing custom handling of printing alerts. Your override code of the `GXHandleAlertEvent` message must match the following formal declaration:

```
OSErr MyHandleAlertEvent (gxStatusRecord *aStatusRecord,
    DialogPtr *aDialogPtr, EventRecord *theEvent, short *itemHit);
```

`aStatusRecord`

A pointer to the status structure.

`aDialogPtr`

The handle to the printing alert box.

`theEvent`

A pointer to the event structure containing the alert event.

`itemHit`

The ID of the dialog item that was selected by the user.

## Printing Messages

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

**DESCRIPTION**

QuickDraw GX sends the `GXHandleAlertEvent` message when a printer driver calls the `GXAlertTheUser` function with a status structure containing a status type of `gxUserAlert`.

The default implementation of this message handles the standard set of status alerts. If this message is forwarded to the universal driver and if it is not a message for the universal driver, the default implementation assumes that the caller has set up a printing alert box and handles the event accordingly.

You do not need to override this message if your alert is one of the standard alerts for which QuickDraw GX provides automatic handling, as described in the chapter “Printer Drivers” in this book. If you want to provide custom alerts, first check the `statusOwner` field of the status structure to make sure that this message is intended for you. If so, perform a total override of this message.

In your override, you need to check the event. If the status event dismisses the printing alert box, fill in the `dialogResult` field of the status structure with a nonzero value to inform the Finder that the dialog box can be removed.

**SPECIAL CONSIDERATIONS**

You never send the `GXHandleAlertEvent` message yourself.

If you handle the event, you must totally override the `GXHandleAlertEvent` message. Otherwise, forward the message so that another message handler can process the event.

**RESULT CODES**

<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**

You can find an example of an override of the `GXHandleAlertEvent` message in Listing 3-18 on page 3-48 in the chapter “Printer Drivers.”

The `GXAlertTheUser` function is described on page 5-18 in the chapter “Printing Functions for Message Overrides.”

The status structure is described on page 4-39.



## GXHandleAlertStatus

---

The Finder or QuickDraw GX may send the `GXHandleAlertStatus` message. You need to override the `GXHandleAlertStatus` message if you want to display status or error dialog boxes that are more complex than those supported by the standard status handling. Your override of the `GXHandleAlertStatus` message must match the following formal declaration:

```
OSErr MyHandleAlertStatus (gxStatusRecord *statRecPtr);
```

`statRecPtr` A pointer to the status structure.

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

### DESCRIPTION

The Finder sends the `GXHandleAlertStatus` message when it receives the status from a background printing process and wishes to alert the user. QuickDraw GX sends the `GXHandleAlertStatus` message when a printer driver calls the `GXAlertTheUser` function with a status structure containing a status type of `gxUserAttention`.

The default implementation of this message expects that the caller wants to display a printing alert box and performs the actions necessary to do so.

You need to override this message if you want to display status dialog boxes or printing alert boxes that are more complex than are supported by this standard status handling. If you want to override this message, you first check the `statusOwner` field of the status structure to make sure that this message is intended for you. If it is intended for you, you are free to manage the dialog box yourself.

### SPECIAL CONSIDERATIONS

You never send the `GXHandleAlertStatus` message yourself.

If you handle the status, perform a total override of the `GXHandleAlertStatus` message. Otherwise, forward the message so that another message handler can process it.

### RESULT CODES

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

### SEE ALSO

The `GXAlertTheUser` function is described on page 5-18 in the chapter “Printing Functions for Message Overrides” in this book.

The status structure is described on page 4-39.

## GXHandleAlertFilter

---

You need to override the `GXHandleAlertEvent` message if you are providing custom handling of printing alerts. Your override code of the `GXHandleAlertEvent` message must match the following formal declaration:

```
OSErr MyGXHandleAlertFilter (gxJob aJob, gxStatusRecord *aStatus,
                             DialogPtr aDialog, EventRecord *aEventRecord,
                             short *itemHit, Boolean *returnImmed);
```

<code>aJob</code>	The job object that is associated with the dialog box.
<code>aStatus</code>	A pointer to the status record that describes the alert.
<code>aDialog</code>	A pointer to information about the dialog box that is displaying the alert information.
<code>aEventRecord</code>	A pointer to information about the event that is being handled.
<code>itemHit</code>	A pointer to the ID of the item in the dialog box in which the event occurred.
<code>returnImmed</code>	On return, a Boolean value that is <code>true</code> if the filter procedure completely handled the event (no further event processing needs to be done) and <code>false</code> if the filter procedure did not completely handle the event.
<i>function result</i>	An error code. The value <code>noErr</code> indicates that the operation was successful.

### DESCRIPTION

QuickDraw GX sends the `GXHandleAlertFilter` message when a user-initiated event occurs in a printing alert box.

The default implementation of this message handles events that occur in the the printing alert box. If you have customized the printing alert box, you need to handle those events as necessary.

In your override, you need to check the event. If the status event dismisses the printing alert box, set the `returnImmed` parameter value to `true`.

### SPECIAL CONSIDERATIONS

You never send the `GXHandleAlertFilter` message yourself.

## Printing Messages

If you handle the event, you must totally override the `GXHandleAlertFilter` message. Otherwise, forward the message so that another message handler can process the event.

**RESULT CODES**

<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.

**Finder Menu Messages**

You can use Finder menu messages if you need to provide users with setup options or interaction through the Finder's Printing menu. The Finder sends these messages during printing from the Finder.

**GXGetDTPMenuList**

The Finder sends the `GXGetDTPMenuList` message when it displays the Printing menu for a particular desktop printer. You can override the `GXGetDTPMenuList` message if you need to add menu items to the Printing menu. Your override of the `GXGetDTPMenuList` message must match the following formal declaration:

```
OSErr MyGetDTPMenuList (MenuHandle aMenu);
```

`aMenu`            A handle to the menu to which you are adding items.

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

**DESCRIPTION**

The default implementation of this message provides the initial list of Printing menu items.

You can override this message if you need to add menu items to the Printing menu. You can use the `AppendMenu` function to add your items.

**SPECIAL CONSIDERATIONS**

You never send the `GXGetDTPMenuList` message yourself.

You must forward the `GXGetDTPMenuList` message. Forward the message first and then add your menu items.

## Printing Messages

## 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

For more information on desktop printers and the Printing menu, see *Inside Macintosh: QuickDraw GX Printing*.

## GXDTPMenuSelect

---

The Finder sends the `GXDTPMenuSelect` message when the user chooses a menu item in the Printing menu for a specific printing device. You must override the `GXDTPMenuSelect` message if you have added items to the Printing menu. Your override of the `GXDTPMenuSelect` message must match the following formal declaration:

```
OSErr MyDTPMenuSelect (short id);
```

**id**                The ID of the menu item chosen from the Printing menu by the user.

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

## DESCRIPTION

You need to override the `GXDTPMenuSelect` message if you add items to the Printing menu. If the item selected is less than or equal to the number of items you have added, handle that item. If the value is positive, you need to subtract your total number of items from the item value before forwarding the message.

If the value of the `id` parameter is less than or equal to the number of items that you added in your override of the `GXGetDTPMenuList` message, then you need to handle the selection.

If the `id` parameter is greater than the number of items that you added in the `GXGetDTPMenuList` message, then you need to subtract this total from the `id` parameter and forward `GXDTPMenuSelect` to enable the next message handler to utilize the message.

**SPECIAL CONSIDERATIONS**

You never send the `GXDTPMenuSelect` message yourself.

If you override the `GXDTPMenuSelect` message, you also need to override the `GXGetDTPMenuList` message.

If you handle the chosen menu item, do not forward the `GXDTPMenuSelect` message. Otherwise, you must forward it.

**RESULT CODES**

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

**SEE ALSO**

The `GXGetDTPMenuList` message is described in the previous section.

For more information about desktop printers and the Printing menu, see *Inside Macintosh: QuickDraw GX Printing*.

## Summary of Printing Messages

---

### Data Types for Printer Drivers

---

#### The Print-to-File Structure

```
struct gxPrintDestinationRec {
    Boolean    printToFile;    /* true if printing to file */
    FSSpec     fSpec;         /* print file FSSpec */
    Boolean    includeFonts;   /* true if system fonts included in file */
    Str31      fileFormat;     /* file format name */
};

typedef struct gxPrintDestinationRec gxPrintDestinationRec,
*gxPrintDestinationPtr, **gxPrintDestinationHdl;
```

#### The Printing Buffer Structure

```
struct gxPrintingBuffer {
    long       size;          /* size of buffer in bytes */
    long       userData;      /* ID assigned by driver or extension for buffer */
    char       data[1];       /* array of data bytes, size bytes long */
};

typedef struct gxPrintingBuffer gxPrintingBuffer;
```

#### The Page Information Structure

```
struct gxPageInfoRecord {
    long       docPageNum;    /* number of page being printed */
    long       copyNum;       /* copy number being printed */
    Boolean    formatChanged; /* whether format changed from prev */
    Boolean    pageChanged;   /* whether contents changed */
    long       internalUse;    /* private */
};

typedef struct gxPageInfoRecord gxPageInfoRecord;
```

## Constants and Data Types for Macintosh Printing Manager Compatibility

---

### The Universal Print Structure

```

struct gxUniversalPrintRecord {
    short    prVersion;        /* print record version */

                                /* prInfo subrecord... */
    short    appDev;           /* device kind, always 0 */
    short    appVRes;          /* application vert resolution */
    short    appHRes;          /* application horiz resolution */
    Rect     appPage;          /* page size, in application resolution */
    Rect     appPaper;         /* paper rectangle [offset from appPage] */

                                /* prStl subrecord... */
    short    devType;          /* device type, always 0 (wDev) */
    short    pageV;            /* page height in 120ths of inch */
    short    pageH;            /* page width in 120ths of inch */
    char     filler;           /* unused */
    char     feed;             /* feed mode */

                                /* prInfoPT subrecord... */
    short    devKind;          /* device kind, always 0xA900 */
    short    devVRes;          /* device vertical resolution */
    short    devHRes;          /* device horizontal resolution */
    Rect     devPage;          /* device page size */

                                /* prXInfo subrecord... */
    short    actualCopies;      /* actual number of copies for this job */
    short    options;          /* options for this device */
    short    reduction;        /* reduction/enlargement factor */
    char     orientation;       /* orientation of paper */

    char     qualityMode;       /* type of quality mode */
    char     firstTray;         /* type of first feed tray */
    char     remainingTray;     /* type of remaining feed tray */

    char     coverPage;        /* type of cover page */
    char     headMotion;        /* type of head motion */
    char     saveFile;          /* type of save file */

    char     userCluster1;      /* unused */
    char     userCluster2;      /* unused */
    char     userCluster3;      /* unused */
}

```

## Printing Messages

```

/* prJob subrecord... */
short    firstPage;    /* number of the first page */
short    lastPage;     /* number of the last page */
short    copies;       /* # of copies, always 1 */
char     reserved1;    /* always true, unused */
char     reserved2;    /* always true, unused */
PrIdleProcPtr
        pIdleProc;     /* pointer to the idle proc */
Ptr       pFileName;   /* pointer to the spool filename */
short    fileVol;      /* spool file volume reference number */
char     fileVers;     /* file version, must be 0 */
char     reserved3;    /* always 0 */

short    printX[19];   /* internal use */
};

typedef struct gxUniversalPrintRecord gxUniversalPrintRecord,
*gxUniversalPrintRecordPtr, **gxUniversalPrintRecordHdl;

```

**Feed Mode Options for the Universal Print Structure**

```

enum {
    gxAutoFeed      = 0,      /* use automatic paper feeding */
    gxManualFeed    = 1      /* use manual paper feeding */
};

```

**Print Options for the Universal Print Structure**

```

enum {
    gxPreciseBitmap    = 0x0001,    /* tall adjust(IW), precise bitmap(LW) */
    gxBiggerPages      = 0x0002,    /* no gaps(IW), larger print area(LW) */
    gxGraphicSmoothing = 0x0004,    /* graphics smoothing for LW */
    gxTextSmoothing    = 0x0008,    /* text smoothing for SC */
    gxFontSubstitution  = 0x0010,    /* font substitution */
    gxInvertPage       = 0x0020,    /* black/white image inversion */
    gxFlipPageHoriz    = 0x0040,    /* flip page image horizontally */
    gxFlipPageVert     = 0x0080,    /* flip page image vertically */
    gxColorMode        = 0x0100,    /* use color printing */
    gxBidirectional    = 0x0200,    /* use bidirectional printing */
    gxUserFlag0        = 0x0400,    /* user flag 0 */
    gxUserFlag1        = 0x0800,    /* user flag 1 */
    gxUserFlag2        = 0x1000,    /* user flag 2 */
    gxReservedFlag0    = 0x2000,    /* reserved flag 0 */
};

```



## Printing Messages

```

    gxReservedFlag1      = 0x4000,    /* reserved flag 1 */
    gxReservedFlag2      = 0x8000    /* reserved flag 2 */
};

```

**Paper Orientation Options for the Universal Print Structure**

```

enum {
    gxPortraitOrientation      = 0,    /* use portrait orientation */
    gxLandscapeOrientation     = 1,    /* use landscape orientation */
    gxAltPortraitOrientation   = 2,    /* use rotated portrait orientation */
    gxAltLandscapeOrientation  = 3     /* use rotated landscape orientation */
};

```

**Print Quality Modes for the Universal Print Structure**

```

enum {
    gxBestQuality              = 0,     /* use best-quality printing */
    gxFasterQuality            = 1,     /* use medium-quality printing */
    gxDraftQuality             = 2      /* use draft-quality printing */
};

```

**Paper Tray Selections for the Universal Print Structure**

```

enum {
    gxFirstTray      = 0,      /* use first paper-feed tray */
    gxSecondTray     = 1,      /* use second paper-feed tray */
    gxThirdTray      = 2       /* use third paper-feed tray */
};

```

**Cover Page Options for the Universal Print Structure**

```

enum {
    gxNoCoverPage      = 0,      /* don't print a cover page */
    gxFirstPageCover   = 1,      /* print cover page before first page */
    gxLastPageCover    = 2       /* print cover page after last page */
};

```

**Print-Head Motions for the Universal Print Structure**

```

enum {
    gxUnidirectionalMotion = 0,    /* use one-directional print-head motion */
    gxBidirectionalMotion  = 1     /* print in both directions */
};

```

**File Save Types for the Universal Print Structure**

```
enum {
    gxNoFile          = 0,          /* not saving print job to file */
    gxPostScriptFile  = 1          /* save print job to PostScript file */
};
```

**Constants and Data Types for the Raster Imaging System**

---

**Raster Offscreen Structure**

```
struct gxOffscreenRec {
    short      numberOfPlanes;      /* the number of planes */
    Handle     offscreenStorage;    /* handle to the image data */
    gxOffscreenPlaneRec
        thePlanes[1];              /* array of planes to draw in */
};
```

```
typedef struct gxOffscreenRec gxOffscreenRec, *gxOffscreenPtr,
**gxOffscreenHdl;
```

**Raster Offscreen Plane Structure**

```
struct gxOffscreenPlaneRec {
    gxViewPort    theViewPort;      /* view port for offscreen plane */
    gxViewDevice  theDevice;        /* view device for offscreen plane */
    gxViewGroup   theViewGroup;     /* view group for offscreen plane */
    gxShape       theBitmap;        /* shape object for offscreen bitmap */
    gxBitMap      theBits;          /* actual bitmap data for this area */
};
```

```
typedef struct gxOffscreenPlaneRec gxOffscreenPlaneRec;
```

**Raster Offscreen Setup Structure**

```
struct gxOffscreenSetupRec {
    short      width;                /* width in pixels of raster */
    short      minHeight;            /* minimum height in pixels */
    short      maxHeight;            /* maximum height in pixels */
    Fixed      ramPercentage;        /* maximum % of RAM to use for raster */
    long       ramSlop;              /* amount of RAM to leave free */
    short      depth;                /* depth in bits of each plane */
    gxMapping  vpMapping;            /* mapping for offscreen view ports */
    gxMapping  vdMapping;            /* mapping for offscreen view devices */
};
```

## Printing Messages

```

short      planes;           /* number of planes to allocate */
gxPlaneSetupRec
           planeSetup[4];    /* parameters for each plane */
};

```

```
typedef struct gxOffscreenSetupRec gxOffscreenSetupRec;
```

**Raster Offscreen Plane Setup Structure**

```

struct gxPlaneSetupRec {
    gxRasterPlaneOptions
        planeOptions;        /* options for this plane */
    gxHalftone    planeHalftone; /* halftone values for this plane */
    gxColorSpace  planeSpace;    /* color space for this plane */
    gxColorSet    planeSet;      /* color set for this plane */
    gxColorProfile planeProfile; /* color profile for this plane */
};

```

```
typedef struct gxPlaneSetupRec gxPlaneSetupRec;
```

**Raster Plane Options**

```

enum {
    /* default value: driver allocates bits & creates halftone values */
    gxDefaultOffscreen      = 0x00000000,,
    /* driver does not call the GXSetViewPortHalftone function */
    gxDontSetHalftone       = 0x00000001,
    /* driver calls GXSetViewPortDither function with gxDotType dithering */
    gxDotTypeIsDitherLevel  = 0x00000002
};

```

```
typedef long gxRasterPlaneOptions;
```

**Raster Package Bitmap Structure**

```

struct gxRasterPackageBitmapRec {
    gxBitmap    *bitmapToPackage; /* pointer to bitmap containing the data */
    unsigned short startRaster;    /* raster where packaging begins */
    unsigned short colorBand;      /* color pass of this package */
    Boolean      isBandDirty;      /* whether there are any non-white */
                                   /* bits in this band */
    Rect         dirtyRect;        /* area containing nonwhite bits*/
};

```

```
typedef struct gxRasterPackageBitmapRec gxRasterPackageBitmapRec;
```

**Raster Imaging System Structure**

```

struct gxRasterImageDataRec {
    /* setup values */
    gxRasterRenderOptions    renderOptions;    /* rendering options */
    Fixed                    hImageRes;        /* horizontal resolution */
    Fixed                    vImageRes;        /* vertical resolution */
    short                    minBandSize; /* minimum band size in pixels */
    short                    maxBandSize; /* maximum band size in pixels */
    gxRectangle              pageSize;         /* size of page in pixels */
    short                    currentYPos;     /* position on page */
    gxRasterPackageRec       packagingInfo; /* raster package structure */
    Boolean                  optionsValid; /* true if options specified */
    gxRasterPackageControlsRec packageControls; /* raster package controls */
    gxOffscreenSetupRec      theSetup;        /* offscreen setup info */
};

typedef struct gxRasterImageDataRec gxRasterImageDataRec,
*gxRasterImageDataPtr, **gxRasterImageDataHdl;

```

**Raster Render Options**

```

enum {
    gxDefaultRaster      = 0x00000000, /* default options */
    gxDontResolveTransferModes
                                = 0x00000001, /* 0=resolve xfer modes, 1=don't */
    gxRenderInReverse    = 0x00000002, /* traverse image in reverse */
    gxOnePlaneAtATime    = 0x00000004, /* render each plane separately */
    gxSendAllBands       = 0x00000008 /* send even empty bands */
};

typedef long gxRasterRenderOptions;

```

**Raster Package Structure**

```

struct gxRasterPackageRec {
    Ptr        bufferSize;    /* buffer size of packaging */
    short      colorPasses;   /* number of color passes */
    short      headHeight;    /* height of print head in pixels */
    short      numberPasses;  /* number of passes per headheight */
    short      passOffset;    /* offset between passes, in pixels */
    gxRasterPackageOptions
        packageOptions; /* packaging options */
};

```

## Printing Messages

```
typedef struct gxRasterPackageRec gxRasterPackageRec, *gxRasterPackagePtr,
**gxRasterPackageHdl;
```

**Raster Package Options**

```
enum {          /* bit fields in gxRasterPackageOptions */
    gxSendAllColors    = 0x00000001, /* send even clean bands */
    gxInterlaceColor   = 0x00000002, /* ribbon contamination */
    gxOverlayColor     = 0x00000004, /* no ribbon problem */
    gxUseColor         = (gxInterlaceColor|gxOverlayColor)
};

typedef long gxRasterPackageOptions;
```

**Constants and Data Types for the Vector Imaging System**

---

**Vector Halftone Structure**

```
struct gxVHalftoneRec {
    gxColorSpace      halftoneSpace; /* color space for device */
    gxHalftoneCompRec halftoneComps[4]; /* array of halftone structures */
    long              penIndexForBW; /* pen index for drawing 1-bit */
                                /* deep black&white bitmaps */
};

typedef struct gxVHalftoneRec gxVHalftoneRec;
```

**Vector Halftone Component Structure**

```
struct gxVHalftoneCompRec {
    Fixed    angle; /* angle to halftone at-this value must
                    be 0, 45, 90, or 135 */
    long     penIndex; /* index of the pen to use for this component */
};

typedef struct gxVHalftoneCompRec gxVHalftoneCompRec;
```

**Vector Shape Structure**

```
struct gxVectorShapeDataRec {
    gxVectorShapeOptions
        shapeOptions; /* options to control shape handling */
    long  maxPolyPoints; /* max polygon points device can support */
    Fixed shapeError; /* the allowed deviation from original shape */
};
```

## Printing Messages

```

    Fixed textSize;          /* delimits filled from outlined text */
    Fixed frameSize;        /* delimits filled from stroked shapes */
};

```

```
typedef struct gxVectorShapeDataRec gxVectorShapeDataRec;
```

**Vector Shape Options**

```

enum {
    gxUnidirectionalFill    = 0x00000001, /* driver needs to generate scan
                                           lines in 1 direction only */
    gxAlsoOutlineFilledShape= 0x00000002 /* driver also needs to draw the
                                           outlines of filled shapes */
};

```

```
typedef long gxVectorShapeOptions;
```

**Vector Imaging System Structure**

```

struct gxVectorImageDataRec { /* vector imaging system structure */
    gxVectorRenderOptions  renderOptions; /* options for vector rendering */
    Fixed                  devRes;        /* resolution of device */
    gxTransform            devTransform; /* transform object for device */
    gxColorSet             clrSet;        /* set of colors on device */
    gxColor                bgColor;      /* background color */
    gxVHaltoneRec          halftoneInfo; /* halftone info for color */
    gxPenTableHdl          hPenTable;    /* complete list of device pens */
    gxRectangle            pageRect;     /* dimensions of the page */
    gxVectorShapeDataRec   shapeData;    /* info on how to render shapes */
};

```

```

typedef struct gxVectorImageDataRec gxVectorImageDataRec,
*gxVectorImageDataPtr, **gxVectorImageDataHdl;

```

**Vector Render Options**

```

enum {
    gxColorSort            = 0x00000001, /* must use this for pen plotters */
    gxATransferMode        = 0x00000002, /* driver needs to resolve xfer modes */
    gxNoOverlap            = 0x00000004, /* driver needs to produce
                                           non-overlapping output */
    gxAColorBitmap         = 0x00000008, /* driver needs to produce color bitmap
                                           output */
    gxSortbyPenPos         = 0x00000010, /* driver needs to draw shapes in same
                                           order as in input */
};

```

## Printing Messages

```

                                order as pens in pen table */
gxPenLessPlotter=0x00000020,    /* output device is raster printer or
                                plotter */
gxCutterPlotter= 0x00000040,    /* output device has a cutter */
gxNoBackGround = 0x00000080    /* shapes that map to background color
                                are not sent to the device */
};

typedef long gxVectorRenderOptions;

```

**Vector Pen Table Structure**

```

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

typedef struct gxPenTable gxPenTable, *gxPenTablePtr, **gxPenTableHdl;

```

**Vector Pen Table Entry Structure**

```

struct gxPenTableEntry {
    Str31      penName;      /* name of the pen */
    gxColor    penColor;     /* color that's part of the color set */
    Fixed      penThickness; /* size of the pen */
    short      penUnits;     /* the units in which pen size is defined */
    short      penPosition;  /* pen position in the carousel; if pen is
                                not loaded, value is kPenNotLoaded (-1) */
};

typedef struct gxPenTableEntry gxPenTableEntry;

```

**Vector Pen Units**

```

enum {
    gxDeviceUnits = 0,
    gxMMUnits     = 1,
    gxInchesUnits = 2
};

```

## Constants and Data Types for the PostScript Imaging System

---

### PostScript Imaging System Structure

```
struct gxPostScriptImageDataRec {
    short      languageLevel; /* PostScript language level */
    gxColorSpace devCSpace;    /* the printer's color space */
    gxColorProfile devCProfile; /* the printer's color profile */
    gxPostScriptRenderOptions
        renderOptions; /* rendering options */
    long      pathLimit; /* maximum number of points in a path */
    short     gsaveLimit; /* maximum gsave's allowed */
    short     opStackLimit; /* maximum number of objects in stack */
    scalerStreamFontFlag
        fontType; /* stream type to use when downloading a
                  font */
    long      printerVM; /* the amount of printer memory */
    long      reserved0; /* unused */
};
```

```
typedef struct gxPostScriptImageDataRec gxPostScriptImageDataRec,
*gxPostScriptImageDataPtr, **gxPostScriptImageDataHandle;
```

### PostScript Render Options

```
enum {
    /* driver needs to convert binary data to 7-bit ASCII values */
    gxNeedsAsciiOption          = 0x00000001,
    /* driver needs to issue PostScript comments */
    gxNeedsCommentsOption       = 0x00000002,
    /* driver needs to calculate bounding box values */
    gxBoundingBoxesOption       = 0x00000004,
    /* driver needs to generate non-device-specific Postscript output */
    gxPortablePostScriptOption   = 0x00000008,
    /* driver needs to use Level 2 device-independent color when
       printing to a Level 2 output device */
    gxUseLevel2ColorOption       = 0x00000080
};
```

```
typedef long gxPostScriptRenderOptions;
```



**PostScript Glyphs Structure**

```

struct gxPrinterGlyphsRec {
    gxFont      theFont;    /* the font */
    long        nGlyphs;    /* how many glyphs it contains */
    gxFontPlatform platform; /* the platform of the font */
    gxFontScript script;    /* the script code of the font */
    gxFontLanguage language; /* the language code of the font */
    long        vmUsage;    /* amount of printer memory needed for font */
    unsigned long glyphBits[1]; /* the glyph data */
};

typedef struct gxPrinterGlyphsRec gxPrinterGlyphsRec;

```

**PostScript Procedure Set List Structure**

```

struct gxProcSetListRec {
    Signature    clientid;    /* unique client ID */
    OSType       controlType; /* resource type of procedure set */
    short        controlid;   /* resource ID of procedure set */
    OSType       dataType;    /* data type of procedure set */
    long         reserved0;
};

typedef struct gxProcSetListRec gxProcSetListRec, *gxProcSetListPtr,
**gxProcSetListHdl;

```

**PostScript Query Results**

```

enum {
    gxPrinterOK          = 0,    /* printer is initialized and ok */
    gxInitializePrinter  = 1,    /* printer needs initialization */
    gxFilePrinting       = 2,    /* printing to file */
    gxResetPrinter       = 128   /* printer needs resetting */
};

```

**User Interface Constants and Data Types**

---

**The Panel Information Structure**

```

struct gxPanelInfoRecord {
    gxPanelEvent panelEvt; /* the event */
    short panelResId;      /* resource ID of current 'ppnl' res */
    DialogPtr pDlg;        /* pointer to dialog */
};

```

## Printing Messages

```

EventRecord *theEvent; /* pointer to event */
short itemHit;          /* actual item number of event */
short itemCount;        /* number of items before your items */
short evtAction;         /* the action that will occur after
                           this event is processed */
short errorStringId;     /* 'STR ' ID of error string */
gxFormat theFormat;      /* the current format */
void *refCon;            /* refCon from gxPanelSetupRecord */
};

```

```
typedef struct gxPanelInfoRecord gxPanelInfoRecord;
```

**Panel Events**

```

enum {
    gxPanelNoEvt      = (gxPanelEvent) 0,      /* no event */
    gxPanelOpenEvt     = (gxPanelEvent) 1,      /* panel is about to open */
    gxPanelCloseEvt    = (gxPanelEvent) 2,      /* panel is about to close */
    gxPanelHitEvt      = (gxPanelEvent) 3,      /* user has selected item */
    gxPanelActivateEvt = (gxPanelEvent) 4,      /* panel has been activated */
    gxPanelDeactivateEvt = (gxPanelEvent) 5,    /* panel has been deactivated */
    gxPanelIconFocusEvt = (gxPanelEvent) 6,     /* focus has changed to icons */
    gxPanelPanelFocusEvt = (gxPanelEvent) 7,    /* focus has changed to panel */
    gxPanelFilterEvt   = (gxPanelEvent) 8,      /* panel event needs to be
                                                filtered */
    gxPanelCancelEvt   = (gxPanelEvent) 9,      /* panel has been canceled */
    gxPanelConfirmEvt  = (gxPanelEvent) 10,     /* panel has been confirmed */
    gxPanelDialogEvt   = (gxPanelEvent) 11,     /* panel event to be handled
                                                by the dialog box handler */
    gxPanelOtherEvt    = (gxPanelEvent) 12,     /* an OS event has occurred
                                                in the panel */
    gxPanelUserWillConfirmEvt
                        = (gxPanelEvent) 13     /* user has selected confirm */
};

```

```
typedef long gxPanelEvent;
```

**Panel Responses**

```

enum {
    gxPanelNoResult      = 0, /* no result from panel */
    gxPanelCancelConfirmation = 1, /* user confirmed panel, but panel
                                    handler discovered an error */
};

```

## Printing Messages

```
};
```

```
typedef long gxPanelResult;
```

**Panel Event Actions**

```
enum {
    gxOtherAction      = 0,          /* current item doesnt change after event */
    gxClosePanelAction = 1,          /* panel is closed after event */
    gxCancelDialogAction = 2,        /* dialog box is canceled after event */
    gxConfirmDialogAction = 3        /* dialog box is confirmed after event */
};
```

**Parse Range Results**

```
enum {
    gxRangeNotParsed  = (gxParsePageRangeResult) 0,    /* not parsed yet */
    gxRangeParsed      = (gxParsePageRangeResult) 1,    /* successful parse */
    gxRangeBadFromValue = (gxParsePageRangeResult) 2,    /* the "from page" */
                                                         /* value is invalid */
    gxRangeBadToValue  = (gxParsePageRangeResult) 3      /* the "to page" */
                                                         /* value is invalid */
};
```

```
typedef short gxParsePageRangeResult;
```

**The Status Structure**

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

```
typedef struct gxStatusRecord gxStatusRecord;
```

## Printing Messages

**The Manual Feed Structure**

```

struct gxManualFeedRecord {
    Boolean    canAutoFeed; /* whether driver can switch to auto feed */
    Str31      paperTypeName; /* name of paperType to feed */
};

typedef struct gxManualFeedRecord gxManualFeedRecord;

```

**The Display Structure**

```

struct gxDisplayRecord { /* for GXWriteStatusToDTPWindow message */
    Boolean    useText; /* true if displaying text, false for picture */
    Handle     hPicture; /* handle for the picture to display
                        (unimplemented & reserved for future use) */
    Str255     theText; /* the text to display */
};

typedef struct gxDisplayRecord gxDisplayRecord;

```

Printing Messages

---

**Storage Messages**

```

OSErr GXInitialize          (void);
OSErr GXShutDown            (void);
OSErr GXFetchTaggedData     (ResType aResType, short id,
                             Handle *aHandle, Signature owner);

```

**Print Object Messages**

```

OSErr GXDefaultJob          (void);
OSErr GXDefaultFormat       (gxFormat aFormat);
OSErr GXDefaultPaperType    (gxPaperType aPaperType);
OSErr GXDefaultPrinter      (gxPrinter aPrinter);
OSErr GXDefaultDesktopPrinter
                             (Str31 dtpName);

```

**Application Messages**

```

OSErr GXStartJob            (StringPtr docName, long pageCount);
void GXCleanupStartJob      (void);
OSErr GXFinishJob           (void);
OSErr GXJobIdle             (void);

```

## Printing Messages

```

OSErr GXStartPage          (gxFormat aFormat, long numViewPorts,
                             gxViewPort *viewPortList);

void GXCleanupStartPage    (void);

OSErr GXFinishPage         (void);

void GXPrintPage           (gxFormat aFormat, gxShape aPage);

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

OSErr GXParsePageRange     (StringPtr fromString, StringPtr toString,
                             gxParsePageRangeResult *result);

```

**Paper-Handling Messages**

```

OSErr GXDoesPaperFit       (gxTrayIndex whichTray, gxPaperType paper,
                             Boolean *fits);

```

**Color Profile Messages**

```

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

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

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

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

```

**Spooling Messages**

```

OSErr GXCreateSpoolFile    (FSSpecPtr aFSSpecPtr,
                             long createOptions, gxSpoolFile *aSpoolFile);

OSErr GXSpoolPage          (gxSpoolFile aSpoolFile, gxFormat aFormat,
                             gxShape aShape);

OSErr GXSpoolData          (gxSpoolFile aSpoolFile, Ptr data,
                             long *length);

OSErr GXSpoolResource      (gxSpoolFile aSpoolFile, Handle aResource,
                             ResType aType, short id);

OSErr GXCompleteSpoolFile  (gxSpoolFile aSpoolFile);

```

**Despooling Messages**

```

OSErr GXCountPages         (gxSpoolFile aSpoolFile, long *numPages);

```

## Printing Messages

```

OSError GXDespoolPage      (gxSpoolFile aSpoolFile,
                             long pageNum, gxFormat aFormat,
                             gxShape *aShape, Boolean *formatChanged);

OSError GXDespoolData      (gxSpoolFile aSpoolFile,
                             Ptr data, long *length);

OSError GXDespoolResource  (gxSpoolFile aSpoolFile,
                             ResType aType, short id,
                             Handle *aResource);

OSError GXExamineSpoolFile (gxSpoolFile aSpoolFile);

OSError GXCcloseSpoolFile  (gxSpoolFile aSpoolFile,
                             long closeOptions);

```

**Dialog Box Messages**

```

OSError GXPrintingEvent    (EventRecord *anEventRecord,
                             Boolean filterEvent);

OSError GXJobDefaultFormatDialog
                             (gxDialogResult *aDialogResult);

OSError GXFormatDialog     (gxFormat aFormat, StringPtr title,
                             gxDialogResult *aDialogResult);

OSError GXJobPrintDialog   (gxDialogResult *aDialogResult);

OSError GXHandlePanelEvent (gxPanelInfoRecord *aPanelInfoRecord
                             gxPanelResult *panelResult);

OSError GXFilterPanelEvent (gxPanelInfoRecord *aPanelInfoRecord;
                             Boolean *returnImmed);

```

**Universal Imaging Messages**

```

OSError GXJobStatus        (gxStatusRecord *aStatusRecord);

OSError GXCaptureOutputDevice (Boolean capture);

OSError GXImageJob         (gxSpoolFile aSpoolFile,
                             long *closeOptions);

OSError GXCreateImageFile  (FSSpecPtr aFSSpecPtr,
                             long imageFileOptions, long *fileReference);

OSError GXSetupImageData   (void *imageData);

OSError GXImageDocument    (gxSpoolFile aSpoolFile, void *imageData);

OSError GXImagePage        (gxSpoolFile aSpoolFile, long pageNumber,
                             gxFormat aFormat, void *imageData);

OSError GXRenderPage       (gxFormat aFormat, gxShape aShape,
                             gxPageInfoRecord *aPageInfoRecord,
                             void *imageData);

```

**Raster Imaging Messages**

```

OSErr GXRasterDataIn      (gxOffscreenHdl offScreen,
                           gxRectangle *bandRectangle,
                           gxRectangle *dirtyRectangle);

OSErr GXRasterLineFeed    (short *lineFeedSize,
                           Ptr buffer, unsigned long *bufferPos,
                           gxRasterImageDataHdl imageData);

OSErr GXRasterPackageBitmap (gxRasterPackageBitmapRec *whattoPackage,
                             Ptr buffer, unsigned long *bufferPos,
                             gxRasterImageDataHdl imageData);

```

**PostScript Imaging Messages**

```

OSErr GXPostScriptQueryPrinter
                           (long *queryResult);

OSErr GXPostScriptInitializePrinter
                           (void);

OSErr GXPostScriptResetPrinter
                           (void);

OSErr GXPostScriptExitServer (void);

OSErr GXPostScriptGetStatusText
                           (Handle statusTextHdl);

OSErr GXPostScriptGetPrinterText
                           (Handle printerTextHdl);

OSErr GXPostScriptScanStatusText
                           (Handle statusTextHdl);

OSErr GXPostScriptScanPrinterText
                           (Handle printerTextHdl);

OSErr GXPostScriptGetDocumentProcSetList
                           (gxProcSetListHdl procSetListHdl,
                           gxPostScriptImageDataHandle hImageData);

OSErr GXPostScriptDownloadProcSetList
                           (gxProcSetListHdl procSetListHdl,
                           gxPostScriptImageDataHandle hImageData);

OSErr GXPostScriptGetPrinterGlyphsInformation
                           (gxPrinterGlyphsRec *glyphPtr);

OSErr GXPostScriptStreamFont (gxFont fontRef, gxScalerStream *stream);

OSErr GXPostScriptDoDocumentHeader
                           (gxPostScriptImageDataHandle hImageData);

OSErr GXPostScriptDoDocumentSetup
                           (gxPostScriptImageDataHandle hImageData);

OSErr GXPostScriptDoDocumentTrailer
                           (gxPostScriptImageDataHandle hImageData);

```

## Printing Messages

```

OSErr GXPostScriptDoPageSetup
    (gxFormat aFormat, long pageIndex,
     gxPostScriptImageDataHandle hImageData);

OSErr GXPostScriptSelectPaperType
    (gxPaperType aPaperType, long pageIndex,
     gxPostScriptImageDataHandle hImageData);

OSErr GXPostScriptDoPageTrailer
    (gxPostScriptImageDataHandle hImageData);

OSErr GXPostScriptEjectPage (gxPaperType aPaperType, long pageIndex,
    long copiesCount, short erasePage,
    gxPostScriptImageDataHandle hImageData);

OSErr GXPostScriptProcessShape
    (gxShape aShape, long count,
     gxTransform list[]);

```

**Vector Imaging Messages**

```

OSErr GXVectorPackageShape (gxShape aShape, long penIndex);

OSErr GXVectorLoadPens      (gxPenTableHdl penTable,
    long *shapeCounts, Boolean *penTableChanged);

OSErr GXVectorVectorizeShape (gxShape aShape, long penIndex,
    gxVectorShapeDataRec *aVectorShapeDataRec);

```

**Device Communications Messages**

```

OSErr GXOpenConnection      (void);

OSErr GXOpenConnectionRetry (ResType commType, void *commData,
    Boolean *pRetry, OSERR saveErr);

void GXCleanupOpenConnection (void);

OSErr GXCloseConnection     (void);

OSErr GXStartSendPage       (gxFormat pageFormat);

void GXCleanupStartSendPage (void);

OSErr GXFinishSendPage      (void);

OSErr GXBufferData          (Ptr data, long length, long bufferOptions);

OSErr GXWriteData           (Ptr data, long length);

OSErr GXDumpBuffer          (gxPrintingBuffer *aPrintingBuffer);

OSErr GXFreeBuffer          (gxPrintingBufferPtr *aPrintingBuffer);

OSErr GXFinishSendPlane     (void);

OSErr GXCheckStatus         (Ptr data, long length, short statusType,
    Signature owner);

OSErr GXGetDeviceStatus     (Ptr cmdData, long cmdSize,
    Ptr responseData, long *responseSize,
    Str255 termination);

```



**Compatibility Messages**

```

OSErr GXPrOpenDoc          (THPrint aTHPrint, TPPrPort *aTPPrPort);
OSErr GXPrCloseDoc        (TPPrPort aTPPrPort);
OSErr GXPrOpenPage        (TPPrPort aTPPrPort, TPRect aTPRect,
                           Point resolution);
OSErr GXPrClosePage       (TPPrPort aTPPrPort);
OSErr GXPrintDefault      (THPrint aTHPrint);
OSErr GXPrStlDialog       (THPrint aTHPrint, Boolean *aBoolean);
OSErr GXPrJobDialog       (THPrint aTHPrint, Boolean *aBoolean);
OSErr GXPrStlInit        (THPrint aTHPrint, TPPrDlg *aTPPrDlg);
OSErr GXPrJobInit        (THPrint aTHPrint, TPPrDlg *aTPPrDlg);
OSErr GXPrDlgMain        (THPrint aTHPrint,
                           PDlgInitProcPtr aPDlgInitProcPtr,
                           Boolean *aBoolean);
OSErr GXPrValidate       (THPrint aTHPrint, Boolean *aBoolean);
OSErr GXPrGeneral        (Ptr aPtr);
OSErr GXPrJobMerge       (THPrint aTHPrint1, THPrint aTHPrint2);
OSErr GXConvertPrintRecordFrom
                           (THPrint aTHPrint);
OSErr GXConvertPrintRecordTo (THPrint aTHPrint);
OSErr GXPrintRecordToJob  (THPrint aTHPrint, gxJob aJob);

```

**Finder Dialog Box Messages**

```

OSErr GXWriteStatusToDTPWindow
                           (gxStatusRecord aStatusRecord,
                           gxDisplayRecord aDisplayRecord);
OSErr GXInitializeStatusAlert
                           (gxStatusRecord *statRecPtr,
                           DialogPtr *dPtr);
OSErr GXHandleAlertEvent  (gxStatusRecord *aStatusRecord,
                           DialogPtr *aDialogPtr, EventRecord *theEvent,
                           short *itemHit);
OSErr GXHandleAlertStatus (gxStatusRecord *statRecPtr);
OSErr GXHandleAlertFilter (gxJob ajob, gxStatusRecord *aStatus,
                           DialogPtr aDialog, EventRecord *aEventRecord,
                           short *itemNum, Boolean *returnImmed);

```

**Finder Menu Messages**

```

OSErr GXGetDTPMenuList   (MenuHandle aMenu);
OSErr GXDTPMenuSelect    (short id);

```

