

This chapter describes the printing functions that you can call only from within your overrides of the printing messages. These functions perform a variety of useful operations. You need to call many of these functions when implementing a QuickDraw GX printer driver, and you can call many of them when writing a QuickDraw GX printing extension.

Before reading this chapter, you need to understand how to write QuickDraw GX printing extensions and printer drivers, as described in the chapters “Printing Extensions” and “Printer Drivers” in this book. You also need to understand the QuickDraw GX printing architecture, which is described in *Inside Macintosh: QuickDraw GX Printing*.

This chapter begins with a brief overview of the QuickDraw GX printing functions and then describes how to use them to perform tasks related to implementing a printing extension or printer driver. The section “Printing Functions Reference” beginning on page 5-13, provides a description for each of the functions and for each of the data types that is used with the functions.

About the Printing Functions

You can only call the printing functions from within your implementations of printing message overrides. You call these functions to perform certain tasks while overriding a printing message. These functions are implemented by QuickDraw GX, just like the QuickDraw GX functions that you use to display shapes.

▲ WARNING

You can only call the functions that are described in this chapter from within an override of a QuickDraw GX printing message. If you call one of these functions in any other context, your program will crash. ▲

In its implementation of some of these functions, QuickDraw GX sends a corresponding printing message to the handlers in the message chain. For example, you can call the `GXJobIdle` function, the implementation of which sends the `GXJobIdle` message to the top of the message chain. Each message handler can then respond to the message as required.

The printing cleanup functions work in a slightly different manner. The QuickDraw GX implementation of each of the printing cleanup functions sends a corresponding printing message; however, the message is sent to the next handler in the chain, not to the top of the chain. For example, when you call the `GXCleanupOpenConnection` function, QuickDraw GX sends the `GXCleanupOpenConnection` message to the handler after your printing extension or printer driver in the message chain. QuickDraw GX implements the printing cleanup functions in this way to ensure that cleanup messages are sent to the same handlers in the same order as were the original printing messages.

Using the Printing Functions

You use the printing functions to perform actions from within your overrides of printing messages. You generally use these functions from within message overrides in a printer driver, though you can also use them in printing extension code.

You use the printing functions to

- display status information and printing alert boxes to the user
- manage which paper type is assigned to each paper tray on a printer
- store and access information in a desktop printer
- communicate which imaging options your driver provides to applications
- access data that your printing extension or printer driver needs from the job collection and Macintosh system software
- clean up after an error occurs during the processing of certain printing messages

This section provides general information about using the functions for each of these tasks.

Displaying Status Information and the Printing Alert Boxes

QuickDraw GX defines two types of user feedback that you can provide from your printing extension or printer driver: status information and printing alert boxes. You display status information in a user's desktop printer window when the information does not require any user response. You display printing alert boxes when you need the user to respond prior to continuing the printing process.

You use the `GXReportStatus` function to display status information to the user in the desktop printer window. You use the `GXAlertTheUser` function to display printing alert boxes to the user. Both of these functions are described in the section “Displaying Status Information and the Printing Alert Boxes” beginning on page 3-41 in the chapter “Printer Drivers.”

QuickDraw GX also provides the printing alert (`'plrt'`) resource for defining printing alert boxes. The use of printing alert boxes is also described in the section “Displaying Status Information and the Printing Alert Boxes” beginning on page 3-41 in the chapter “Printer Drivers.” You use the `GXGetPrintingAlert` function to display a printing alert box.

You can use the `GXPrintingAlert` function to create a printing alert box when you have not defined a resource for this purpose. This function takes as parameters all of the values that you would specify in a printing alert resource, draws the printing alert box, and displays it with any informative text you supply. You get the same results as you would by calling the `GXGetPrintingAlert` function with the ID of a printing alert that you defined in a resource.

Managing Paper Trays and Their Paper Types

QuickDraw GX allows the user to assign specific paper types to specific paper trays on a printer, as described in *Inside Macintosh: QuickDraw GX Printing*. You can use the tray-handling functions from within your message overrides to set or determine the paper type associated with each tray.

The paper trays on a printer are either configured or not. If one or more of the trays on the printer has a specific paper type assigned to it, then the trays are configured. If none of the trays has a specific paper type assigned to it, then the trays are unconfigured. You can call the `GXGetTrayMapping` function to discover if the printer with which your driver is communicating has trays that are configured.

If you need to cycle through the trays on a printer, you can call the `GXCountTrays` function to discover how many trays the printer has. For example, if you need to find which tray contains a certain paper type, you can call `GXCountTrays` and use the value it returns to limit your search.

If you want to determine if a certain paper type is in one of the paper trays, you can first call `GXGetTrayMapping`. If it tells you that the trays are not configured, then you don't need to look any farther. If it tells you that the trays are configured, you can call `GXCountTrays` to determine how many trays there are, and then loop through the trays.

You can determine the name of each tray by calling the `GXGetTrayName` function.

You can set the paper type that is associated with a tray by calling the `GXSetTrayPaperType` function, and you can determine which paper type is associated with a tray by calling the `GXGetTrayPaperType` function.

Listing 5-1 shows how the LaserWriter printer driver fills in the paper-type names in the Printing menu by looping through the paper trays.

Listing 5-1 Looping through the paper trays

```
for (i = iEnvelopePopUp; i <= iLargePopUp; ++i)
{
    GetDItem(dPtr, i, &theKind, &theHandle, &box);
    SetCtlMin((ControlHandle) theHandle, 1);
    SetCtlMax((ControlHandle) theHandle, CountMItems(theMenu));
    SetCtlValue((ControlHandle) theHandle, 1);

    /* if you don't have it, hide it*/
    if (
        ( (i == iEnvelopePopUp) && (!(options & kEnvelopeMask)) )
        || ( (i == iLargePopUp) && (!(options & k500SheetMask)) )
    )
        HideDItem(dPtr, i);
    else
    {
```

Printing Functions for Message Overrides

```

anErr = GXGetTrayPaperType(i-iEnvelopePopUp+1, paper);
if (anErr == noErr)
{
    Str31    paperName, menuString;
    short    j;

    GXGetPaperTypeName(paper, paperName);

    for (j = CountMItems(theMenu); j > 1; --j)
    {
        GetItem(theMenu, j, menuString);
        if (IUCompString(menuString, paperName) == 0)
        {
            SetCtlValue((ControlHandle) theHandle, j);
            break;
        }
    }
}

/*
    If you don't have the resource, it is not an error;
    it means that you have a fresh tray setup.
*/
if (anErr == resNotFound)
    anErr = noErr;
nrequire(anErr, FailedGetTrayPaperType);
}

```

Storing and Accessing Data Associated With a Desktop Printer

Applications can store and access information in the data collections that are associated with a specific desktop printer. This allows you to preserve information with the desktop printer that your printing extension or printer driver can use when processing any document.

For example, you could discover that a printer has a color ribbon and store that fact with the desktop printer so that you could report it to the application. Or if the user has set a special option for the printer in the Printing menu, you might want to store that option with the desktop printer so that your driver could use it during printing.

Printing Functions for Message Overrides

QuickDraw GX provides two functions for interacting with the desktop printer data from within your message overrides. You use the `GXWriteDTPData` function to store data with the desktop printer, and you use the `GXFetchDTPData` function to access data that has been stored with the desktop printer.

Listing 5-2 shows the ImageWriter II printer driver's function for determining if the printer has a color ribbon. This function uses the `GXFetchDTPData` function and returns `true` if the printer does have a color ribbon.

Listing 5-2 Accessing the desktop printer data

```
Boolean PrinterHasColorRibbon(gxPrinter thePrinter)
{
    Boolean          hasColor = true;
    Str32            deviceName;
    OSErr            anErr;
    ImageWriterConfigHandle configHandle;

    /*
     * If not formatting to a particular device, then assume
     * that there is a color ribbon, to allow for the widest
     * range of color spaces.
     */
    GXGetPrinterName(thePrinter, deviceName);
    if (deviceName[0] != 0)
    {
        /*
         * If formatting for a particular device, then assume that
         * there is not a color ribbon because that is more common.
         */
        hasColor = false;

        anErr = GXFetchDTPData(deviceName, kImageWriterConfigType,
                               kImageWriterConfigID, &(Handle)configHandle);
        if (anErr == noErr)
        {
            hasColor = (**configHandle).hasColorRibbon;
            DisposHandle((Handle) configHandle);
        }
    }

    return(hasColor);
}
```

Providing Application Imaging Options

Application programs can provide the user with various print-quality and speed options. If you are writing a QuickDraw GX printing extension or printer driver, you may need to communicate to the application what options you support. These options come in two forms: printer view devices, which define the kinds of imaging that your driver can perform, and job format modes, which define the kinds of data interface that your printer driver supports.

You create a view device for each kind of imaging that your printer can support. You might provide a black-and-white view device and a color view device. You might provide a low-resolution view device and a high-resolution view device. You call the `GXAddPrinterViewDevice` function to add each of your supported view devices to the list of available view devices that the application can examine.

For example, the ImageWriter II printer driver sets a black-and-white 144 dpi view device and an 8-color 144 dpi view device in its override of the `GXDefaultPrinter` message. The code for this override is shown in Listing 3-4 on page 3-24 in the chapter “Printer Drivers.” QuickDraw GX view devices are described in *Inside Macintosh: QuickDraw GX Objects*.

Your printer driver also needs to communicate with the application about which job format modes the user can request. The job format mode enumeration is described on page 5-14. Each mode defines a kind of data with which your driver can operate, including graphics data, text-only data, and PostScript data. Job format modes are described in *Inside Macintosh: QuickDraw GX Printing*.

As a driver developer, you need to find out which job format modes the application provides to the user and then specify one of those modes as your preferred mode. You call the `GXGetAvailableJobFormatModes` function to determine which job format modes the application is providing. You then pick one of those modes as your preferred mode and call the `GXSetPreferredJobFormatMode` function to communicate your preference to the application. The preferred mode is the mode that gets set when the user selects direct-mode printing.

For example, in its override of the `GXJobDefaultFormatDialog` message, the ImageWriter II printer driver loops through the job format modes supported by the application to determine if the application supports text mode (for faster printing). If it does, the driver makes that the preferred mode. The code for this override is shown in Listing 3-7 on page 3-29 in the chapter “Printer Drivers.” Job format modes are described in *Inside Macintosh: QuickDraw GX Printing*.

Accessing Driver Data

Your driver sometimes needs to access data that belongs to the current print job. You can use the `GXGetJob` function to obtain a reference to the current job object.

You sometimes also need to access resource data that belongs to your driver. You can retrieve the resource file reference number for your driver by calling the `GXGetMessageHandlerResFile` function. You can also access your resource data by

Printing Functions for Message Overrides

overriding the `GXFetchTaggedData` message, which is described on page 4-45 in the chapter “Printing Messages” in this book.

Interfacing With the Chooser

When the user selects your driver in the Chooser, the Chooser sends certain Chooser messages to you by calling the `Device` function, which is the interface for the package ('PACK') resource. You need to provide a version of this function to provide Chooser support for your driver. The package resource and its use are described in *Inside Macintosh: More Macintosh Toolbox*.

QuickDraw GX provides the `GXHandleChooserMessage` function, which takes care of most of the work of handling the Chooser package code for you. What you need to do is create a `Device` function, take appropriate action for messages that you need to handle, and pass the parameters on to the `GXHandleChooserMessage` function. Listing 5-3 shows the ImageWriter II printer driver implementation of the `Device` function.

Listing 5-3 The `Device` function for the ImageWriter II printer driver

```
pascal OSErr Device(short message, short caller,
                    StringPtr objName, StringPtr zoneName,
                    ListHandle theList, long p2)
{
    OSErr          anErr = noErr;
    extern Str31    gDriverName;
    StringPtr      pDriverName = &gDriverName;
    extern gxJob    gJob;
    gxJob          *pJob = &gJob;

    /* start up QuickDraw GX to begin with */
    if (message == initializeMsg)
    {
        FCBPBBRec   pb;

        /* determine the driver name */
        pb.ioCompletion = nil;
        pb.ioNamePtr = pDriverName;
        pb.ioVRefNum = 0;
        pb.ioRefNum = CurResFile();
        pb.ioFCBIndx = 0;
        anErr = PBGetFCBInfo(&pb, false);

        *pJob = nil;
    }
}
```

Printing Functions for Message Overrides

```

        if (anErr == noErr)
        {
            GXEnterGraphics();
            anErr = GXInitPrinting();
            if (anErr == noErr)
                anErr = GXNewJob(pJob);
        }
    }

    /* let the system handle the choosing */
    if (anErr == noErr)
        anErr = GXHandleChooserMessage(*pJob, pDriverName, message,
                                       caller, objName, zoneName, theList, p2);

    /* shut down QuickDraw GX when done */
    if ( (message == terminateMsg) && (p2 == terminateMsg) )
    {
        if (*pJob != nil)
            GXDisposeJob(*pJob);
        GXExitPrinting();
        GXExitGraphics();
    }

    return(anErr);
}

```

The only Chooser messages that the ImageWriter II printer driver needs to do something special with are `initializeMsg` and `terminateMsg`. When the Chooser sends `initializeMsg`, the ImageWriter II driver initializes QuickDraw GX printing before passing the message along. And when the Chooser sends the `terminateMsg` message, the ImageWriter II driver terminates QuickDraw GX printing after passing the message along. The Chooser messages are described in *Inside Macintosh: Devices*.

Using the Message Cleanup Functions

When you override a message in your printing extension or printer driver, you often forward the message to other message handlers so that they can add their operations to the execution of the message.

Some QuickDraw GX message handlers allocate storage or modify state variables in response to certain printing messages. If something goes wrong in the processing of those messages, these handlers need to reverse their actions. The messages that require this reversing of actions, or cleanup, are the `GXOpenConnection`, `GXStartJob`,

Printing Functions for Message Overrides

GXStartPage, and GXStartSendPage messages, which are described in the chapter “Printing Messages” in this book.

If you forward one of these messages from within your override code and then detect an error, you need to call the cleanup function. Each cleanup function sends a cleanup message to the handlers, starting with the handler immediately below your extension or driver in the chain. This ensures that each handler receives the message only once and in the appropriate order. The cleanup functions are: GXCleanupOpenConnection, GXCleanupStartJob, GXCleanupStartPage, and CleanupGXStartSendPage.

Listing 5-4 shows the ImageWriter II printer driver override of the GXOpenConnection message. This code uses the nrequire macro for exception handling and calls the GXCleanupOpenConnection function if an error occurs during the processing of the GXOpenConnection message. The nrequire macro is described in the chapter “Printer Drivers” in this book.

Listing 5-4 Calling the GXCleanupOpenConnection function

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

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

    /* then, bring the configuration file up to date */
    anErr = UpdateConfiguration();
    nrequire(anErr, UpdateConfiguration);

    return(noErr);

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

GXOpenConnection:

    return(anErr);
}
```

If an error occurs during the execution of the GXOpenConnection message, this override function calls the GXCleanupOpenConnection function.

Segmenting Your Driver Code

The smallest code segment that you can have with the messaging architecture consists of the code required to override a single message. However, in some cases you might want to have multiple code segments for a single message. For example, you might need a number of functions to override a message like `GXRenderPage`, which performs a lot of work.

QuickDraw GX provides a segment handler that you can use to divide your message override code into multiple segments. The segment handler allows you to construct a special 32-bit dispatch selector, which allows you to access functions in other code segments. The dispatch selector contains the segment's resource ID and the offset in that segment to the start of your function.

The `PRINTINGDISPATCH` macro constructs the dispatch selector and dispatch code for you, which allows you to call the function just as you would any other function. All that you need to do is define the macro. `PRINTINGDISPATCH` assumes that you start your segment with a long-aligned jump table, just like the jump tables that you use for your printing message overrides, as described in the section "The Jump Table" beginning on page 2-9 in the chapter "Printing Extensions." It also assumes that each entry in the table is 4 bytes long.

You define the dispatch macro with the segment ID and routine selector. For example, if your code is the third routine in a segment (it is at offset 12 in the segment) in a printer driver ('pdvr') resource with an ID of 2, you define your macro as shown here:

```
OSErr MyRenderingRoutine (long param1, Ptr param2)
    = PRINTINGDISPATCH(2, 3);
```

All segment dispatches must return a value of type `OSErr`. If your function does not generate errors, you must still declare it to return an `OSErr` value and have the function return the constant `noErr`. You can then call your function as usual and also get free type-checking, as in this example:

```
anErr = MyRenderingRoutine(p1, p2);    /* free type-checking */
```

You can call across segments by constructing your own 32-bit selector and calling the `GXPrintingDispatch` function directly. This method does not give you free type-checking and makes your code more complicated. However, the results are identical. For example, you could call the `MyRenderingRoutine` function using the following method:

```
#define kMyRenderRoutineSelector 0x0002000C
/* no type checking for the following call */
anErr = GXPrintingDispatch(kMyRenderRoutineSelector, p1, p2);
```

The `GXPrintingDispatch` function is described on page 5-38.

Printing Functions Reference

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

The “Constants and Data Types” section shows the data structures and enumerated types that are used with the printing functions.

The “Functions” section describes the interface and functionality of each function that you can call only from within your overrides of the printing messages.

Constants and Data Types

This section describes the constants and data types that you use with the printing functions. Each of the types in this section is used as the type of a parameter to one or more of the printing functions.

Several data types that are used with the printing functions are described elsewhere. Notably, the `gxJob` and `gxPaperType` data types are described in *Inside Macintosh: QuickDraw GX Printing*.

Tray Index Type

The tray index type is used to designate a specific paper tray on a printer.

```
typedef long gxTrayIndex;
```

Tray Mapping Modes

The tray-mapping mode constants define the current tray-mapping mode of the printer. The `GXGetTrayMapping` function returns a value of a tray-mapping type, which you can use to determine if you need to loop through the trays to find information.

```
enum {
    gxDefaultTrayMapping      = (gxTrayMapping) 0,
    gxConfiguredTrayMapping   = (gxTrayMapping) 1
};
```

```
typedef long gxTrayMapping;
```

Constant descriptions

`gxDefaultTrayMapping`

None of the paper trays has a specific paper type assigned to it.

`gxConfiguredTrayMapping`

At least one of the paper trays has a specific paper type assigned to it.

Job Format Mode Table

The `GXGetAvailableJobFormatModes` function uses a job format mode table structure to communicate with the application about which job format modes are supported.

```
struct gxJobFormatModeTable {
    long          numModes;
    gxJobFormatMode  modes[1];
};

typedef struct gxJobFormatMode Table gxJobFormatModeTable,
*gxJobFormatModeTablePtr, **gxJobFormatModeTableHdl;
```

Field descriptions

<code>numModes</code>	The number of the entries in the <code>modes</code> array that follows.
<code>modes</code>	An array of mode constants. This array can be of any length. The constants are defined in the job format modes enumeration, which is defined in the next section.

Job Format Modes

The job format modes enumeration defines the constants used in the `modes` field of the job format mode table, which is described in the previous section. The constants define the current data mode of the printer.

```
enum {
    gxGraphicsJobFormatMode = (gxJobFormatMode) 'grph',
    gxTextJobFormatMode     = (gxJobFormatMode) 'text',
    gxPostScriptJobFormatMode= (gxJobFormatMode) 'post'
};

typedef OSType gxJobFormatMode;
```

Constant descriptions

<code>gxGraphicsJobFormatMode</code>	Standard graphics job-format mode. In this mode, a printer driver supports QuickDraw GX graphics and typography.
<code>gxTextJobFormatMode</code>	Text job-format mode. In this mode, a printer driver supports native printer fonts and horizontal and vertical lines.
<code>gxPostscriptJobFormatMode</code>	PostScript job-format mode. In this mode, a printer driver supports the PostScript language.

The Panel Setup Structure

The panel setup structure, of data type `gxPanelSetupRecord`, is passed to the `GXSetupDialogPanel` function when the user displays a dialog box.

```
struct gxPanelSetupRecord {
    gxPrintingPanelKind    panelKind;
    short                  panelResId;
    short                  resourceRefNum;
    void                   *refCon;
};

typedef struct gxPanelSetupRecord gxPanelSetupRecord;
```

Field descriptions

<code>panelKind</code>	The kind of program that is using this panel. This value is one of the constants defined in the printing panel kinds enumeration, which is described in the next section.
<code>panelResId</code>	The resource ID of the panel ('ppnl ') resource for the dialog box panel.
<code>resourceRefNum</code>	The resource file reference number for the panel.
<code>refCon</code>	A reference constant for use by the creator of the panel.

Printing Panel Kinds

The printing panel kinds enumeration provides constants for use in the `panelKind` field of the panel setup structure, which is described in the previous section.

```
enum {
    gxApplicationPanel= (gxPrintingPanelKind) 0,
    gxExtensionPanel   = (gxPrintingPanelKind) 1,
    gxDriverPanel      = (gxPrintingPanelKind) 2
};

typedef long gxPrintingPanelKind;
```

Constant descriptions

<code>gxApplicationPanel</code>	A panel created for an application.
<code>gxExtensionPanel</code>	A panel created for a printing extension.
<code>gxDriverPanel</code>	A panel created for a printer driver.

Functions

This section describes the functions that you can call only from within your overrides of the printing messages. The functions are divided into the following categories:

- The status and alert display functions are used to display feedback to the user during the process of printing a document.
- The tray-handling functions are used to manage which paper is assigned to each paper tray on the printer.
- The desktop-printer data functions are used to store and access resource information in a desktop printer.
- The dialog box panel function is used to manage the addition of panels to a print dialog box.
- The application imaging options functions are used to communicate with the application regarding the imaging options provided by your printing extension or printer driver.
- The printing control functions are used to manage contextual information about your printing extension or printer driver.
- The message cleanup functions are used to perform cleanup tasks when certain printing messages fail.
- The segment control function allows you to segment your version of a printing message override.

Reporting Information to the User

You use the status and alert display functions to report information to the user.

The `GXReportStatus` function displays information that does not require user feedback, such as messages that monitor the current state of the document being printed. Typical status information text strings include “2 pages remaining to print” and “Starting job.”

The `GXAlertTheUser` function displays information to the user in a printing alert box. The user must respond to the printing alert box. Typical printing alert text strings include “Printer is out of paper” and “Printer requires attention: paper jam.”

The `GXGetPrintingAlert` function displays a printing alert box that is defined in a printing alert ('plrt') resource.

The `GXPrintingAlert` function builds a printing alert box from the supplied parameters and displays it on the user's screen.

GXReportStatus

You can use the `GXReportStatus` function to display information to the user in the desktop printer window.

```
OSErr GXReportStatus ( short statusId,  
                      unsigned short statusIndex );
```

`statusId` The ID of the status ('stat') resource that contains the status strings.

`statusIndex` The index of the status string that you want displayed.

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

DESCRIPTION

You call the `GXReportStatus` function to display informational feedback to the user during the printing of a document. You need to specify the ID of the status resource in which the feedback string is stored.

If the information that you want to display requires user attention, you need to call the `GXAlertTheUser` function instead.

RESULT CODES

<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>gxUnknownAlertVersionErr</code>	The printing alert version specified in the resource is not valid.

SEE ALSO

You can find an example of using the `GXReportStatus` function in Listing 3-15 on page 3-42 in the chapter “Printer Drivers.”

You can read about status resources in the section “The Status ('stat') Resource” beginning on page 6-19 in the chapter “Printing Resources.”

You can read about displaying status information in the section “Displaying Status Information and the Printing Alert Boxes” beginning on page 5-4. You can also read about this task in the section “Displaying Status Information and the Printing Alert Boxes” beginning on page 3-41 in the chapter “Printer Drivers.”

GXAlertTheUser

You can use the `GXAlertTheUser` function to display a printing alert box to the user.

```
OSErr GXAlertTheUser (gxStatusRecord *status);
```

`status` A status structure of type `gxStatusRecord`.

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

DESCRIPTION

The `GXAlertTheUser` function displays status information to the user in a printing alert box. The information to be displayed is described in the `status` parameter. The user must explicitly dismiss the printing alert box, and you must monitor the user's actions while the alert box is displayed.

RESULT CODES

<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 using the `GXAlertTheUser` function in Listing 3-15 on page 3-42 and in Listing 3-16 on page 3-44 in the chapter “Printer Drivers.”

A description of how to use the `GXAlertTheUser` function, including an example from the ImageWriter II driver, is found in the section “Displaying Status Information and the Printing Alert Boxes” beginning on page 3-41 in the chapter “Printer Drivers.”

The `gxStatusRecord` structure is described in the section “The Status Structure” beginning on page 4-39 in the chapter “Printing Messages.”

GXGetPrintingAlert

You can use the `GXGetPrintingAlert` function to display a printing alert box defined by a printing alert resource.

```
OSErr GXGetPrintingAlert (short alertResId,
                          ModalFilterProcPtr filterProc, short *itemHit);
```

`alertResId` The ID of the printing alert ('plrt') resource that defines the contents of the alert box.

Printing Functions for Message Overrides

filterProc A pointer to the function to use for filtering events that occur while the alert box is displayed.

itemHit A pointer to the ID of the item that the user selected to dismiss the alert box.

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

DESCRIPTION

The `GXGetPrintingAlert` function displays a printing alert box. The contents of the alert box are defined in the printing alert resource that has the ID specified in the `alertResId` parameter.

User-initiated events that occur while the alert box is displayed are filtered through the function that is specified by the `filterProc` parameter. When the user dismisses the printing alert box by selecting an item, this function returns in the `itemHit` parameter the ID of the item that was selected.

RESULT CODES

<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 printing alert resource is described in the section “The Printing Alert ('plrt') Resource” beginning on page 6-21 in the chapter “Printing Resources”.

You can use the `GXPrintingAlert` function, as described in the next section, to display a printing alert box whose contents are defined as parameters to the function.

GXPrintingAlert

You can use the `GXPrintingAlert` function to create and display a printing alert box. This function creates the alert box from its parameters. The `GXGetPrintingAlert` function, which is described in the previous section, displays a printing alert box that is created from a resource definition.

```
OSErr GXPrintingAlert (short iconId, short txtSize,
                      short defaultTitleNum, short cancelTitleNum,
                      short textLength, Ptr pAlertMsg,
                      StringPtr actionTitle, StringPtr title2,
```

Printing Functions for Message Overrides

```
StringPtr title3, StringPtr msgFont,
ModalFilterProcPtr filterProc, short *itemHit,
StringPtr alertTitle);
```

<code>iconId</code>	The type of icon to display in the printing alert box. Use one of the following constants for the value of this parameter: <code>noIcon</code> , <code>stopIcon</code> , <code>cautionIcon</code> , or <code>noteIcon</code> .
<code>txtSize</code>	The font size of the text to display in the printing alert box. You can use the value <code>defaultSystemSize</code> to indicate that the default text size for the system is to be used.
<code>defaultTitleNum</code>	The type of text string to display on the default button. Use one of the following constants as the value of this parameter: <code>noDefaultTitle</code> , <code>defaultAction</code> , <code>defaultTitle2</code> , or <code>defaultTitle3</code> .
<code>cancelTitleNum</code>	The type of text string to display on the Cancel button. Use one of the following constants as the value of this parameter: <code>noCancelTitle</code> , <code>cancelAction</code> , <code>cancelTitle2</code> , or <code>cancelTitle3</code> .
<code>textLength</code>	The length of the text string that is to be displayed in the printing alert box.
<code>pAlertMsg</code>	A pointer to the message text to display in the printing alert box.
<code>actionTitle</code>	A pointer to the string to display on the action button.
<code>title2</code>	A pointer to the string to display on button 2.
<code>title3</code>	A pointer to the string to display on button 3.
<code>msgFont</code>	A pointer to a string that names the font to use for displaying the text string in the printing alert box.
<code>filterProc</code>	A pointer to the function to use for filtering events that occur while the alert is displayed.
<code>itemHit</code>	On return, the ID of the item that was selected by the user to dismiss the alert box.
<code>alertTitle</code>	The string that is displayed in the title bar of the printing alert box.
<i>function result</i>	An error code. The value <code>noErr</code> indicates that the operation was successful.

DESCRIPTION

The `GXPrintingAlert` function displays a printing alert box, building it from the parameters that you pass instead of from a printing alert ('`plrt`') resource.

Printing Functions for Message Overrides

User-initiated events that occur while the alert box is displayed are filtered through the function that is specified by the `filterProc` parameter. When the user dismisses the alert box by selecting an item, this function returns in the `itemHit` parameter the ID of the item that was selected.

Each of the parameters to this function matches a field in the printing alert resource.

RESULT CODES

<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 printing alert resource is described in the section “The Printing Alert (‘plrt’) Resource” beginning on page 6-21 in the chapter “Printing Resources.”

You can use the `GXGetPrintingAlert` function, which is described on page 5-18, to display a printing alert box that is created from a resource definition.

Managing Paper Trays

You use the tray-handling functions to manage the use of the paper trays on the printer with which your printing extension or printer driver is communicating. Each tray on the printer is named, and each can have a specific paper type assigned to it.

You use the `GXCountTrays` function when you need to loop through the trays on the printer and determine which kind of paper is assigned to each. The `GXCountTrays` function tells you how many trays you need to search.

The `GXGetTrayName` function returns the name of a specific tray on the printer. The `GXGetTrayMapping` function tells you whether or not any of the trays on the printer has been assigned a specific paper type. If not, then you don’t need to loop through them to find an assigned type.

The `GXSetTrayPaperType` function allows you to associate a specific paper type with a specific tray on the printer, and the `GXGetTrayPaperType` function allows you to access the paper-type information that is associated with a specific tray.

GXCountTrays

You can use the `GXCountTrays` function to determine the number of paper trays that are available for your printing extension or printer driver to use.

```
OSErr GXCountTrays (gxTrayIndex *numTrays);
```

Printing Functions for Message Overrides

numTrays A pointer to a value that, on return, contains the number of paper trays that are currently available on the printer.

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

DESCRIPTION

The `GXCountTrays` function calls the `GXFetchTaggedData` function to retrieve the number of paper trays that are currently available on the printer. You call the `GXCountTrays` function to determine how many trays there are when you need to loop through the paper trays to look for information.

RESULT CODES

<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 `GXFetchTaggedData` function is described in *Inside Macintosh: QuickDraw GX Environment and Utilities*.

You can read about managing paper trays in the section “Managing Paper Trays and Their Paper Types” beginning on page 5-5.

GXGetTrayName

You can use the `GXGetTrayName` function to retrieve the name of a specified paper tray on the printer.

```
OSErr GXGetTrayName (gxTrayIndex whichTray, Str31 trayName);
```

whichTray The number of the tray in which you are interested.

trayName On return, the name of the specified tray. This parameter is a string that you have already allocated.

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

DESCRIPTION

The `GXGetTrayName` function fills in the `trayName` string with the name of the specified tray on the printer. This information is retrieved by calling the `GXFetchTaggedData` function.

RESULT CODES

<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 `GXFetchTaggedData` function is described in *Inside Macintosh: QuickDraw GX Environment and Utilities*.

You can read about managing paper trays in the section “Managing Paper Trays and Their Paper Types” beginning on page 5-5.

GXGetTrayMapping

You can use the `GXGetTrayMapping` function to determine the current tray-mapping mode of the printer.

```
OSERR GXGetTrayMapping (gxTrayMapping *trayMapping);
```

`trayMapping`

A pointer to a value that, on return, is the current tray-mapping mode of the printer.

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

DESCRIPTION

The `GXGetTrayMapping` function returns the current tray-mapping mode of the printer in the `trayMapping` parameter. If any of the paper trays has an assigned paper type, the value of `trayMapping` is set to `gxConfiguredTrayMapping`; otherwise, the value of `trayMapping` is set to `gxDefaultTrayMapping`.

You can use this function to determine if you need to loop through the paper trays to find a certain paper type. If the returned value of `trayMapping` is `gxConfiguredTrayMapping`, you know that at least one of the trays contains a specific paper type and that you do need to loop through the trays. If the returned value of `trayMapping` is `gxDefaultTrayMapping`, none of the trays has a specific paper type assigned to it, so you don't need to loop through them.

Printing Functions for Message Overrides

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 `gxTrayMapping` enumeration is described on page 5-13.

You can read about managing paper trays in the section “Managing Paper Trays and Their Paper Types” beginning on page 5-5.

GXSetTrayPaperType

You can use the `GXSetTrayPaperType` function to change the paper-type information for a specific paper tray on the printer.

```
OSErr GXSetTrayPaperType (gxTrayIndex whichTray,
                          gxPaperType paper);
```

whichTray The number of the tray for which you are changing the paper-type information.

paper A reference to a paper-type object that contains information about the paper that is in the specified tray. If you specify `nil` as the value of this parameter, the tray becomes unconfigured.

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

DESCRIPTION

The `GXSetTrayPaperType` function modifies the paper-type information that is associated with the specified tray on the printer. The information that you supply in the `paper` parameter is stored with the desktop printer and is used to facilitate the handling of mismatched paper types on the printer.

RESULT CODES

<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

You can read about managing paper trays in the section “Managing Paper Trays and Their Paper Types” beginning on page 5-5.

GXGetTrayPaperType

You can use the `GXGetTrayPaperType` function to retrieve the paper-type information for a specific paper tray on the printer.

```
OSErr GXGetTrayPaperType (gxTrayIndex whichTray,
                          gxPaperType paper);
```

<code>whichTray</code>	The number of the tray in which you are interested.
<code>paper</code>	A reference to a paper-type object that you have already allocated. On return, this object contains information about the paper that is in the specified tray.

function result An error code. The values `noErr` indicates that the operation was successful. The value `resNotFound` indicates that the tray is not configured and that its paper type is displayed to the user as “Unknown.”

DESCRIPTION

The `GXGetTrayPaperType` function fills in the fields of a paper-type object with information from the desktop printer.

RESULT CODES

<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

You can read about managing paper trays in the section “Managing Paper Trays and Their Paper Types” beginning on page 5-5.

Storing and Accessing Desktop Printer Data

You use the desktop printer data functions to store and access information in the data collections associated with a specific desktop printer. You store information with the desktop printer when you want that information to be accessible across multiple print jobs.

Desktop printers are described in *Inside Macintosh: QuickDraw GX Printing*.

GXWriteDTPData

You can use the `GXWriteDTPData` function to store data in a desktop printer.

```
OSErr GXWriteDTPData (Str31 dtpName, OSType theType,
                     short theID, Handle theData);
```

<code>dtpName</code>	The name of the desktop printer in which the data is to be stored.
<code>theType</code>	The type of the data that you are storing.
<code>theID</code>	The ID of the data that you are storing.
<code>theData</code>	A handle to the data.

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

DESCRIPTION

The `GXWriteDTPData` function stores data in the desktop printer.

You supply the name of the printer in the `dtpName` parameter, the type in the `theType` parameter, and the ID in the `theID` parameter. `GXWriteDTPData` copies the data from the handle specified by the `theData` parameter.

You must call the `DisposeHandle` function on the handle after calling this function.

RESULT CODES

<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 using the `GXWriteDTPData` function in Listing 3-9 on page 3-31 in the chapter “Printer Drivers.”

The `DisposeHandle` function is described in *Inside Macintosh: Memory*.

GXFetchDTPData

You can use the `GXFetchDTPData` function to access data from the desktop printer.

```
OSErr GXFetchDTPData (Str31 dtpName, OSType theType,
                     short theID, Handle *theData);
```

<code>dtpName</code>	The name of the desktop printer from which to retrieve the data.
<code>theType</code>	The type in which you are interested.
<code>theID</code>	The ID in which you are interested.
<code>theData</code>	A pointer to a handle that, on return, references the data.

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

DESCRIPTION

The `GXFetchDTPData` function accesses data from the desktop printer. You can call this function if you need to examine the data. `GXFetchDTPData` accesses the data and returns a handle to it in the `theData` parameter.

The handle that you receive back from this function has been detached, which means that you must call the `DisposeHandle` function on it when you have finished with it.

RESULT CODES

<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 using the `GXFetchDTPData` function in Listing 3-9 on page 3-31 in the chapter “Printer Drivers.”

The `DisposeHandle` function is described in *Inside Macintosh: Memory*.

Adding a Panel to a Print Dialog Box

You use the dialog box panel function, `GXSetupDialogPanel`, to add a panel to a print dialog box.

GXSetupDialogPanel

You can use the `GXSetupDialogPanel` function to add a panel to a print dialog box.

```
OSErr GXSetupDialogPanel (gxPanelSetupRecord *panelRec);
```

`panelRec` A pointer to a panel setup structure.

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

DESCRIPTION

The `GXSetupDialogPanel` function adds a panel, as defined by the information in the panel setup structure, to a print dialog box. You call this function from within your override of the `GXJobPrintDialog`, `GXFormatDialog`, and `GXJobDefaultFormatDialog` messages, before forwarding the message.

RESULT CODES

<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>gxCantAddPanelsNowErr</code>	Panels can only be added to a dialog box when the current driver is switched. This error is generated if a panel addition request is made at any other time.
<code>gxBadxdtlKeyErr</code>	An unknown key value was specified for an item in an extended dialog control resource.
<code>gxXdtlItemOutOfRangeErr</code>	An item referenced by the panel does not belong to the panel.
<code>gxNoActionButtonErr</code>	The action button for the panel is <code>nil</code> .
<code>gxTitlesTooLongErr</code>	The length of the button titles exceeds the maximum width allowed for a printing alert.

SEE ALSO

You can find an example of using the `GXSetupDialogPanel` function in Listing 2-10 on page 2-18 in the chapter “Printing Extensions.”

The `gxPanelSetupRecord` structure is described in the section “The Panel Setup Structure” on page 5-15.

The `GXJobPrintDialog`, `GXFormatDialog`, and `GXJobDefaultFormatDialog` messages are described in the chapter “Printing Messages” in this book.

Working With Application Imaging Options

You use the application imaging-option functions in your printer driver to communicate with the application regarding which imaging options are available for printing documents. These options include which kinds of view devices your driver supports and which job format mode your driver prefers.

GXAddPrinterViewDevice

You can use the `GXAddPrinterViewDevice` function to add a view device to the list of view devices that can be used with a printer.

```
OSErr GXAddPrinterViewDevice (gxPrinter printer,
                              gxViewDevice viewDev);
```

`printer` A reference to a printer object.

`viewDev` The view device to associate with the printer.

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

DESCRIPTION

The `GXAddPrinterViewDevice` function associates a view device with a specified printer. You call this function from your printer driver to provide information to the application about the kinds of imaging that your printer driver supports for the printer. For example, you can make a view device that is 32 bits deep and associate that with the printer to communicate to the application that your printer driver can manage imaging that is 32 bits deep.

This function increments the owner count of the view device, which means that you still need to call the `GXDisposeViewDevice` function after calling this function.

RESULT CODES

<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 using the `GXAddPrinterViewDevice` function in Listing 3-5 on page 3-25 in the chapter “Printer Drivers.”

The `GXDisposeViewDevice` function is described in *Inside Macintosh: QuickDraw GX Objects*.

GXGetAvailableJobFormatModes

You can use the `GXGetAvailableJobFormatModes` function to determine which job format modes the application supports.

```
OSErr GXGetAvailableJobFormatModes (
                                gxJobFormatModeTableHdl *modeTbl);
```

`modeTbl` A pointer to a `gxJobFormatModeTable` structure.

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

DESCRIPTION

You call the `GXGetAvailableJobFormatModes` function to find out which of the job format modes the application supports. The application establishes these modes by calling the `GXSetPreferredJobFormatMode` function.

Your printer driver calls this function and you pick one of the modes that the application supports as the preferred mode for your printer driver. You then call the `GXSetPreferredJobFormatMode` function to establish that mode as the preference for the application.

RESULT CODES

<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 using the `GXGetAvailableJobFormatModes` function in Listing 3-7 on page 3-29 in the chapter “Printer Drivers.”

The `gxJobFormatModeTable` structure is described on page 5-14.

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

GXSetPreferredJobFormatMode

You can use the `GXSetPreferredJobFormatMode` function to define, for the application, which job format mode is the preferred mode for your printer driver.

```
OSErr GXSetPreferredJobFormatMode (gxJobFormatMode mode,
                                Boolean directOnly);
```

Printing Functions for Message Overrides

mode The job format mode that your driver prefers.

directOnly A Boolean value that is `true` if your driver only supports text mode and `false` if not.

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

DESCRIPTION

The `GXSetPreferredJobFormatMode` function establishes which of the job format modes is the preferred mode of a printer driver. If a printer driver supports PostScript, it generally sets `gxPostscriptJobFormatMode` as its preferred mode. If a driver only supports the use of built-in text fonts, it sets `gxTextJobFormatMode` as its preferred mode. Otherwise, a printer driver generally sets `gxGraphicsJobFormatMode` as its preferred mode.

You set the `directOnly` value to `true` if your printer driver does not support graphics mode. For instance, this is the case for a daisy-wheel printer. In this case, you communicate to the application that it cannot send graphics to be printed by your driver by setting the `directOnly` value to `true`.

RESULT CODES

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

`gxPrUserAbortErr` The user has canceled printing.

SEE ALSO

You can find an example of using the `GXSetPreferredJobFormatMode` function in Listing 3-7 on page 3-29 in the chapter “Printer Drivers.”

The job format mode constants are described on page 5-14.

Printing Control Functions

The printing control functions are a collection of miscellaneous functions that you can use within the implementations of your printing message overrides. They include

- the `GXGetJob` function, which you can call to obtain the job object for the current printing job that your printing extension or printer driver is handling
- the `GXGetMessageHandlerResFile` function, which you can call to obtain the resource file reference number for your printing extension or printer driver
- the `GXSpoolingAborted` function, which you can call to determine if the spooling of a document has been aborted
- the `GXJobIdle` function, which you can call to release idle time to other processes

Printing Functions for Message Overrides

- the `GXHandleChooserMessage` function, which you can call to handle the Chooser messages that are sent to your printer driver

GXGetJob

You can use the `GXGetJob` function to retrieve the job object that is currently associated with your printing extension or printer driver.

```
gxJob GXGetJob (void);
```

function result The job object for the current print job.

DESCRIPTION

The `GXGetJob` function returns the current print job as its function result. You can call this function if you need to access the information that is associated with the job object, including its collections.

RESULT CODES

<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.

GXGetMessageHandlerResFile

You can use the `GXGetMessageHandlerResFile` function to retrieve the resource file reference number of your printing extension or printer driver.

```
short GXGetMessageHandlerResFile (void);
```

function result The resource file reference number of your printing extension or printer driver.

DESCRIPTION

The `GXGetMessageHandlerResFile` function returns the resource file reference number for your extension or driver. You can use this function if you need to access data from your resources.

RESULT CODES

<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.

GXJobIdle

You can use the `GXJobIdle` function to release time to other processes while your printing extension or printer driver is performing a computationally intensive task.

```
OSErr GXJobIdle (void);
```

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

DESCRIPTION

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

RESULT CODES

<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.

GXSpoolingAborted

You can use the `GXSpoolingAborted` function to determine if a user has cancelled spooling of a document.

```
Boolean GXSpoolingAborted (void);
```

function result A Boolean value. A value of `true` indicates that the current printing job has been aborted. A value of `false` indicates that it has not been aborted.

DESCRIPTION

The `GXSpoolingAborted` function returns a Boolean value as its function result. This value tells you if spooling of the current document has been aborted.

Printing Functions for Message Overrides

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.

GXHandleChooserMessage

You can use the `GXHandleChooserMessage` function to respond when a user selects a Chooser item for your printer driver.

```
OSErr GXHandleChooserMessage (gxJob *aJob, Str31 driverName,
                               short message, short caller,
                               StringPtr objName, StringPtr zoneName,
                               ListHandle theList, long p2);
```

<code>aJob</code>	A reference to a job object.
<code>driverName</code>	The name of the printer driver; for example, "LaserWriter."
<code>message</code>	The operation that is to be performed.
<code>caller</code>	The caller. A value of 1 indicates the Chooser. Values in the range 0 to 127 are reserved; values outside of this range can be used by applications.
<code>objName</code>	A pointer to other information. The value of this parameter depends on the value of the <code>message</code> parameter.
<code>zoneName</code>	The name of the AppleTalk zone containing the devices in the device list.
<code>theList</code>	A handle to the List Manager list that contains the device choices that are displayed in the device list box.
<code>p2</code>	Other information. The meaning of this parameter depends on the value of the <code>message</code> parameter.

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

DESCRIPTION

When the user interacts with your driver in the Chooser, the Chooser sends messages to your driver. What the Chooser sends depends on information in the package ('PACK') resource that you have defined for your driver. The `GXHandleChooserMessage` function makes it possible for your driver to work with the Chooser without you having to write the package-handling code.

When the Chooser sends a message to your driver, you can call this function to allow QuickDraw GX to handle the Chooser actions for you. You can perform your own actions, depending on which message has been sent, and then pass the information that the Chooser sent to you along to this function, which takes care of the rest for you.

The messages that the Chooser can send to your driver are shown in Table 5-1.

Table 5-1 Messages that the Chooser sends to drivers

Message	Value	Explanation
init	11	The Chooser sends this message to your package when the user selects the icon representing your package in the icon list.
newSel	12	If your device package allows multiple selections, the Chooser sends this message to your packages when the user changes or adds a selection.
fillList	13	The Chooser sends this message when the user selects a device icon.
getSel	14	The Chooser sends this message to determine which entries should be selected. The Chooser does not send this message for serial printers.
select	15	If your device package does not allow multiple selections, the Chooser sends this message to your packages when the user selects a device from the device list.
deselect	16	If your device package does not allow multiple selections, the Chooser sends this message to your packages when the user deselects a device in the device list.
terminate	17	The Chooser sends this message when the user selects a different device icon, closes the Chooser window, or changes zones.
button	18	The Chooser sends this message when the user clicks one of the buttons in the Chooser window.

RESULT CODES

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

SEE ALSO

An example of using the `GXHandleChooserMessage` function is shown in Listing 5-3 on page 5-9.

Inside Macintosh: Devices describes the Chooser messages and package resources in detail.

Handling Error Conditions in a Message Override

You use the message cleanup functions when you encounter an error during the processing of certain printing messages that change the state of your driver. When you call a cleanup function, QuickDraw GX sends a corresponding cleanup message to all of the handlers in the message chain, allowing them to revert to their previous state.

GXCleanupOpenConnection

You can use the `GXCleanupOpenConnection` function to notify QuickDraw GX that it needs to clean up any actions that it took after receiving a `GXOpenConnection` message from your printing extension or printer driver.

```
void GXCleanupOpenConnection (void);
```

DESCRIPTION

You need to call the `GXCleanupOpenConnection` function if you encounter an error in your override of the `GXOpenConnection` message and you have already forwarded the message.

When you call `GXCleanupOpenConnection`, QuickDraw GX sends the `GXCleanupOpenConnection` message, which allows all of the message handlers to reverse any actions that they took in their overrides of the `GXOpenConnection` message, such as allocating storage or modifying state information.

SEE ALSO

The `GXOpenConnection` and `GXCleanupOpenConnection` messages are described in the chapter “Printing Messages.”

An example of using the `GXCleanupOpenConnection` function is shown in Listing 5-4 on page 5-11.

GXCleanupStartJob

You can use the `GXCleanupStartJob` function to notify QuickDraw GX that it needs to clean up any actions that it took after receiving a `GXStartJob` message from your printing extension or printer driver.

```
void GXCleanupStartJob (void);
```

DESCRIPTION

You need to call the `GXCleanupStartJob` function if you encounter an error in your override of the `GXStartJob` message and you have already forwarded the message.

When you call `GXCleanupStartJob`, QuickDraw GX sends the `GXCleanupStartJob` message, which allows all of the message handlers to reverse any actions that they took in their overrides of the `GXStartJob` message, such as allocating storage or modifying state information.

SEE ALSO

The `GXStartJob` and `GXCleanupStartJob` messages are described in the chapter “Printing Messages.”

GXCleanupStartPage

You can use the `GXCleanupStartPage` function to notify QuickDraw GX that it needs to clean up any actions that it took after receiving a `GXStartPage` message from your printing extension or printer driver.

```
void GXCleanupStartPage (void);
```

DESCRIPTION

You need to call the `GXCleanupStartPage` function if you encounter an error in your override of the `GXStartPage` message and you have already forwarded the message.

When you call `GXCleanupStartPage`, QuickDraw GX sends the `GXCleanupStartPage` message, which allows all of the message handlers to reverse any actions that they took in their overrides of the `GXStartPage` message, such as allocating storage or modifying state information.

SEE ALSO

The `GXStartPage` and `GXCleanupStartPage` messages are described in the chapter “Printing Messages.”

GXCleanupStartSendPage

You can use the `GXCleanupStartSendPage` function to notify QuickDraw GX that it needs to clean up any actions that it took after receiving a `GXStartSendPage` message from your printing extension or printer driver.

```
void GXCleanupStartSendPage (void);
```

DESCRIPTION

You need to call the `GXCleanupStartSendPage` function if you encounter an error in your override of the `GXStartSendPage` message and you have already forwarded the message.

When you call `GXCleanupStartSendPage`, QuickDraw GX sends the `GXCleanupStartSendPage` message, which allows all of the message handlers to

Printing Functions for Message Overrides

reverse any actions that they took in their overrides of the `GXStartSendPage` message, such as allocating storage or modifying state information.

SEE ALSO

The `GXStartSendPage` and `GXCleanupStartSendPage` messages are described in the chapter “Printing Messages.”

Segmenting Message Override Code

The `GXPrintingDispatch` function provides you with a means of segmenting your driver code into smaller boundaries than are permitted by the messaging architecture.

GXPrintingDispatch

You can use the `GXPrintingDispatch` function to make calls across segment boundaries.

```
OSErr GXPrintingDispatch (long selector, ...);
```

`selector` The selector for the printing message that you want to dispatch.
`...` The parameters that the message expects.

DESCRIPTION

The `GXPrintingDispatch` function calls a function that is located in a different code segment. You need to do this if the code to implement a single message override is too large to fit into one segment. This function uses the same information as does the `PRINTINGDISPATCH` macro, which provides a simpler technique for calling across segment boundaries. To use the `GXPrintingDispatch` function, you construct your own selector, as described on page 5-12, and use it as a parameter to the function.

SEE ALSO

An example of using the `GXPrintingDispatch` function is shown on page 5-12.

The `PRINTINGDISPATCH` macro method of calling across segments is described in the section “Segmenting Your Driver Code” beginning on page 5-12.

Summary of Printing Functions

Constants and Data Types

Tray Index Type

```
typedef long gxTrayIndex;          /* specifies a paper tray */
```

Tray Mapping Modes

```
enum {
    gxDefaultTrayMapping      = (gxTrayMapping) 0,      /* no tray has a defined
                                                           paper type assigned */
    gxConfiguredTrayMapping = (gxTrayMapping) 1          /* at least 1 tray has a
                                                           paper type assigned */
};
```

```
typedef long gxTrayMapping;
```

Job Format Modes

```
enum {
    gxGraphicsJobFormatMode    = (gxJobFormatMode) 'grph', /* graphics
                                                           and text */
    gxTextJobFormatMode        = (gxJobFormatMode) 'text',  /* text */
    gxPostScriptJobFormatMode  = (gxJobFormatMode) 'post'   /* PostScript */
};
```

```
typedef OSType gxJobFormatMode;
```

Job Format Mode Table

```
struct gxJobFormatModeTable {
    long          numModes;          /* number of entries in modes field */
    gxJobFormatMode modes[1];        /* array of mode constants */
};
```

```
typedef struct gxJobFormatModeTable gxJobFormatModeTable,
*gxJobFormatModeTablePtr, **gxJobFormatModeTableHdl;
```

The Panel Setup Structure

```

struct gxPanelSetupRecord {
    gxPrintingPanelKind    panelKind;        /* kind of program using panel */
    short                  panelResId;       /* resource ID of panel */
    short                  resourceRefNum;   /* resource file refnum of panel */
    void                   *refCon;         /* pointer to panel setup
                                           structure used to build panel */
};

typedef struct gxPanelSetupRecord gxPanelSetupRecord;

```

Printing Panel Kinds

```

enum {
    gxApplicationPanel= (gxPrintingPanelKind) 0, /* an application panel */
    gxExtensionPanel   = (gxPrintingPanelKind) 1, /* printing extension panel */
    gxDriverPanel      = (gxPrintingPanelKind) 2 /* printer driver panel */
};

typedef long gxPrintingPanelKind;

```

Functions

Reporting Information to the User

```

OSErr GXReportStatus      (short statusID, unsigned short statusIndex);
OSErr GXAlertTheUser      (gxStatusRecord *status);
OSErr GXGetPrintingAlert  (short alertResId,
                          ModalFilterProcPtr filterProc, short *itemHit);
OSErr GXPrintingAlert     (short iconId, short txtSize,
                          short defaultTitleNum, short cancelTitleNum,
                          short textLength, Ptr pAlertMsg,
                          StringPtr actionTitle, StringPtr title2,
                          StringPtr title3, StringPtr msgFont,
                          ModalFilterProcPtr filterProc, short *itemHit,
                          StringPtr alertTitle);

```

Managing Paper Trays

```

OSErr GXCountTrays        (gxTrayIndex *numTrays);
OSErr GXGetTrayName       (gxTrayIndex whichTray, Str31 trayName);
OSErr GXGetTrayMapping    (gxTrayMapping *trayMapping);
OSErr GXSetTrayPaperType  (gxTrayIndex whichTray, gxPaperType paper);

```

Printing Functions for Message Overrides

```
OSErr GXGetTrayPaperType      (gxTrayIndex whichTray, gxPaperType paper);
```

Storing and Accessing Desktop Printer Data

```
OSErr GXWriteDTPData          (Str31 dtpName, OSType theType,
                               short theID, Handle theData);
OSErr GXFetchDTPData          (Str31 dtpName, OSType theType,
                               short theID, Handle *theData);
```

Adding a Panel to a Print Dialog Box

```
OSErr GXSetupDialogPanel      (gxPanelSetupRecord *panelRec);
```

Working With Application Imaging Options

```
OSErr GXAddPrinterViewDevice (gxPrinter printer, gxViewDevice viewDev);
OSErr GXGetAvailableJobFormatModes
                               (gxJobFormatModeTableHdl *modeTbl);
OSErr GXSetPreferredJobFormatMode
                               (gxJobFormatMode, Boolean directOnly);
```

Printing Control Functions

```
gxJob GXGetJob                (void);
short GXGetMessageHandlerResFile
                               (void);
Boolean GXSpoolingAborted      (void);
OSErr GXJobIdle                (void);
OSErr GXHandleChooserMessage   (gxJob *aJob, Str31 driverName,
                               short message, short caller,
                               StringPtr objName, StringPtr zoneName,
                               ListHandle theList, long p2);
```

Handling Error Conditions in a Message

```
void GXCleanupOpenConnection (void);
void GXCleanupStartJob        (void);
void GXCleanupStartPage       (void);
void GXCleanupStartSendPage   (void);
```

Segmenting Message Override Code

```
OSErr GXPrintingDispatch      (long selector, ...);
```

