The PrintOMatic Xtra

Version 1.6.5

The PrintOMatic Xtra is the premier printing tool for Macromedia Director and Authorware. PrintOMatic adds a full set of page-layout, text and graphics printing features to Director 5 and later, and Authorware 4 projects on Macintosh and Windows.

PrintOMatic includes commands for accurately specifying the position of any text or graphic element on the page. PrintOMatic documents can contain graphic files from disk, styled text, graphic primitives, cast member bitmaps, and snapshots of the Director stage.

Product Features:

- Generates multi-page layouts with full control over the placement of text and graphic elements on the page.
- Print styled text: any combination of available fonts, sizes, or styles
- Print text, PICT, BMP or EPS files from disk, any portion of the Director stage, or Director cast members.
- Print object-oriented graphic primitives: lines, boxes, ovals and rounded rectangles.
- "Master Page" can contain any combination of text or graphic elements
- Automatic page numbering
- Customizable and hideable Print Progress dialog
- Paper-saving Print Preview feature
- Supports color and landscape-mode printing
- Supports all Macintosh and Windows compatible printers
- Fully compatible with MacOS™, Windows™ 3.1 and Windows™ 95

Topics in this help file:

PrintOMatic Software Updates

PrintOMatic is updated and enhanced frequently. The latest version of PrintOMatic is always available, along with lots of other stuff, from Electronic Ink's web site:

```
   http://www.printomatic.com/
```

Before reporting a bug in the software, please make sure the version you are using is up to date. Registered users of PrintOMatic will get announcements via e-mail when new major versions are released.

Purchasing PrintOMatic

The PrintOMatic Xtra is published by Electronic Ink. A cross-platform copy of PrintOMatic, with a royalty-free, unlimited use license to use PrintOMatic in your multimedia productions costs $300.00 US. You can register your copy of PrintOMatic on-line at the following location:

    http://www.printomatic.com

The PrintOMatic demonstration movie also prints out an order form for the PrintOMatic Xtra, as well as other Electronic Ink Xtras. To order your copy of PrintOMatic, simply fill out this form and FAX, mail, or e-mail it with payment information to:

Electronic Ink
PO Box 3473
Crested Butte, CO 81224

Tel: (970) 349-1747
Fax: (970) 349-9245
Email: `info@printomatic.com`

Using PrintOMatic

The PrintOMatic Xtra can be used in two ways. The first and simplest way to use PrintOMatic is by calling the global `print` command, passing the Director object(s) you would like to print as parameters:

```
print member "illustration" of castLib 1
print "Some example text for printing"
print sprite 1, sprite 5
print castLib "documentation"
```

If all you need to do is print something out quickly and easily, you can stop reading now; that's all there is to it.

Creating an Instance of the PrintOMatic Xtra

The second way of using PrintOMatic, which is much more powerful and customizable than simply using the global `print` command, is to create an **instance** of the Xtra using the `new` command:

```
set doc = new(xtra "PrintOMatic")
if not objectP(doc) then exit
```

An instance of the PrintOMatic Xtra is called a "document object ", also referred to in this documentation as a "document". Be sure to check the validity of the document object using the `objectP()` function after creating it (shown in the example above). Once you have created a document, you can change its attributes, such as the document's name, margins and landscape orientation:

```
setDocumentName doc, "My Document"
setMargins doc, Rect(36,36,36,36)
setLandscapeMode doc, TRUE
```

Master Page

Page 0 (zero) of a document is the document's "master page". The contents of the master page are drawn on every body page of the document, beneath the body page's contents. When you create a new document, the master page is the default page until you create a new page (see Adding Pages, below.) All items added before the first `newPage` command will appear on the document's master page.

A common item to place on the master page is a page number. Use the `setPageNumSymbol` command in combination with a text item on the master page to place a page number on every page of your document:

```
-- place a page number at the bottom right of the page
set pgNumSym = numToChar(166) -- paragraph symbol on mac
setPageNumSymbol doc, pgNumSym
setTextJust doc, "right"
drawText doc, Point(getPageWidth(doc),getPageHeight(doc)), "page"&&pgNumSym
```

Adding Pages

For a PrintOMatic document to be printable, it must contain at least one page. Add new pages to a document using the `newPage` command. A page added using `newPage` becomes the "current page", where all new graphic elements, "frames", and other items are placed. To make a previously added page (including the master page) the "current page", use the `setPage` command.

```
newPage doc -- add a new page
```

```
setPage doc, 0 -- return to the master page
```

PrintOMatic Coordinate System

PrintOMatic uses the same coordinate system as the Director stage, which places (0,0) at the top-left corner. Coordinate values increase towards the bottom and right sides of the page. All drawing coordinates are specified in points, with 72 points to the inch.

All drawing coordinates for objects in PrintOMatic are relative to the page margins, which are set using the `setMargins` command. The drawing area defined by the margins applies to the entire document.

Calling `pageSetup` or `setLandscapeMode` may change the size of the page, thereby changing the coordinate system, as well as the values returned by `getPageWidth` and `getPageHeight`. If you store these values in Lingo variables, be sure to retrieve them again after calling `pageSetup` or `setLandscapeMode`.

Creating Frames

The most common method of placing text or graphics on a PrintOMatic page is to create one or more rectangular "frames" on the page, then append contents to these frames. The contents of multiple "linked" frames will flow from one frame to the next.

For example, to create a two-column layout, create two frames side by side on the page, and link them together so the text and graphics flow from one frame to the next:

```
-- create a new document
set doc = new(xtra "PrintOMatic")
-- get width and height
set w = getPageWidth(doc)
set h = getPageHeight(doc)
-- create a new page
newPage doc
-- create the first column
newFrame doc, Rect(0,0,(w/2)-18,h), FALSE
-- create the second column, linked to the first
newFrame doc, Rect((w/2)+18,0,w,h), TRUE
```

The last parameter in the `newFrame` command specifies whether or not the new frame is linked to the previous frame.

Appending to Frames

Once you have created one or more "frames" to append to, you can add items such as sprites and cast members to the document using the `append` and `appendFile` commands:

```
append doc, member "title" of castLib 1, TRUE
append doc, sprite 1, TRUE
appendFile doc, the pathName&"myFile.eps", FALSE
```

The last “autoAppend” parameter of `append` and `appendFile` calls specifies whether pages are automatically added to the document as content is inserted. Please see the `append` command for a detailed description of the "autoAppend" feature of PrintOMatic.

IMPORTANT NOTE: Text field cast members will be printed with all their fonts and styles intact. However, Rich Text cast members will only be printed **as bitmaps**, because of the way Director stores their data..

However, text *strings* have no inherent style data associated with them. You can use `setTextFont`, `setTextSize` and `setTextStyle` to set the attributes of text strings that you append.

```
   setTextFont doc, "Helvetica"
   setTextSize doc, 10
   setTextStyle doc, "normal"
   append doc, copyrightInfo
```

Drawing Graphic Elements

A second way of placing items on a PrintOMatic page is to explicitly position them as graphic elements in the exact location you want them to appear. You can place pictures, single lines of text, rectangles, rounded rectangles, lines and ovals on the page in this manner. Many of the graphic element drawing routines take a Lingo `Rect` as the second parameter, to specify their position and size. The last parameter to thse calls designates whether the object is filled (`TRUE`) or stroked (`FALSE`):

```
   drawRect doc, Rect(0,0,50,50), TRUE
   drawRoundRect doc, Rect(0,0,50,50), 25, FALSE
   drawOval doc, Rect(100,100,500,500), TRUE
```

The `drawLine` routine takes a starting `Point` and ending `Point` as its parameters:

```
   drawLine doc, Point(0,0), Point(500,0)
```

The `drawPicture` routine can position a bitmapped image on the page using a `Point` or a `Rect` to specify size and location. If you use a `Point`, the image will be positioned at its normal size (72 dpi for cast members) with its top left corner at the specified point:

```
   drawPicture doc, member "illustration", Point(0,100)
```

If you specify the image location using a `Rect`, the image will be scaled to fit at the largest possible size within the `Rect` *without distorting the image*. Unless the provided `Rect` has the exact same proportions as the image, the entire `Rect` may not be filled with the image data:

```
   drawPicture doc, member "illustration", Rect(0,0,500,300)
```

Drawing the Contents of the Stage

The contents of the Director stage can be placed on a PrintOMatic page using the `drawStagePicture` routine. The size and positioning rules for `drawStagePicture` are the same as for `drawPicture`: if a `Point` is specified, the top left corner of the Stage picture will be placed there; if a `Rect` is specified, the stage image will be scaled to fit within the `Rect` *without distortion.*

```
   drawStagePicture doc, Point(0,0)
   drawStagePicture doc, Rect(0,0,the width of the stage, the height of the
stage)
```

The `drawStagePicture` routine can print a cropped portion of the stage by specifying the area you want to capture after specifying the image's position on the page:

```
   drawStagePicture doc, Rect(0,0,100,100), Rect(50,50,150,150)
```

Finally, you can capture the stage picture from Director's off-screen buffer by adding a `TRUE` to the end of any of the above specified forms of `drawStagePicture`. Capturing from the offscreen buffer will prevent the contents of any windows or MIAWs in front of the Stage from being captured along with the stage image.

```
drawStagePicture doc, Point(0,0), Rect(50,50,100,100), TRUE
```

Printing and Print Preview

When you have appended all the elements you want to print to the document, show the user the job setup dialog, and print the document if `doJobSetup` returns `TRUE`. If you don't want the user to see the job setup dialog, omit the call to doJobSetup.

```
if doJobSetup(doc) then print doc
```

Finally, dispose of the document by setting its value equal to zero. This will release all the memory taken up by the PrintOMatic document.

```
set doc = 0
```

Other Routines

PrintOMatic contains a number of other routines that are not described in the sections above. Please consult the full message list for a complete listing of all the commands supported by the PrintOMatic Xtra.

PrintOMatic Message List

This is the full list of commands supported by the PrintOMatic Xtra:

```
-- CREATE/DESTROY/RESET A DOCUMENT
new object
forget object
reset object
--
-- DOCUMENT/JOB SETUP
doPageSetup object
doJobSetup object
--
-- DOCUMENT ATTRIBUTES
setDocumentName object, string name
setLandscapeMode object, boolean landscape
getLandscapeMode object
setMargins object, rect margins
getMargins object
setPrintableMargins object
getPageWidth object
getPageHeight object
getPaperWidth object
getPaperHeight object
--
-- CREATE/SET PAGES
newPage object -- returns page number
setPage object, int pageNumber
--
-- TEXT/GRAPHIC ATTRIBUTES
setTextFont object, string fontName
setTextSize object, int pointSize
setTextStyle object, string styleCodes
setTextJust object, string justification
setTextLineSpacing object, int spacing
setColor object, int red, int green, int blue
setGray object, int graylevel
setLineWeight object, int pointSize
--
-- GRAPHIC ELEMENTS
drawRect object, rect bounds, boolean filled
drawLine object, point start, point end
drawRoundRect object, rect bounds, int cornerRadius, boolean filled
drawOval object, rect bounds, boolean filled
drawText object, string text, point location
drawPicture object, *
drawStagePicture object, *
--
-- CREATE FRAMES AND APPEND CONTENTS
newFrame object, rect bounds, boolean linkedToPrevious
append object, * any
appendFile object, * fileName
getInsertionPoint object
--
-- CUSTOMIZE THE PROGRESS BOX
setProgressMsg object, string message
```

```
   setProgressPict object, * pictCastMember
   setProgressLoc object, point location
   clearProgressLoc object
   --
   -- PRINT OR PREVIEW
   * printPreview *
   * print *
   * printStage *
   --
   -- MISCELLANEOUS
   hideMessages object, boolean hide
   setPageNumSymbol object, string symbol
   + getVersion object
   + register object, string serialNumber
   + setLowMemLimits object, globalHeap, localHeap
   --
   -- MACINTOSH-ONLY ROUTINES
   * printToPictFiles *
   draw1bitStagePicture object, *
   loadPageSetup object, string fileName, int resourceID
   savePageSetup object, string fileName, string fileType, string fileCreator,
int resourceID
```

Creating, Destroying and Resetting Documents

The following commands are used to create and reset PrintOMatic documents:

`new`	creates a new document
`forget`	**never** call this directly
`reset`	resets a document to defaults

To destroy a document, **never** call the `forget` method of an Xtra explicitly. Instead, use the following syntax to explicitly get rid of a PrintOMatic document when you are done using it:

```
set doc = 0
```

new

Syntax: `set doc = new(xtra "PrintOMatic")`

The new command is used to create a new instance of the PrintOMatic Xtra. The newly created instance is called a **document object**, since it represents a printable "document": a collection of items that are printed together in a single print job by PrintOMatic.

Once the document has been created, its settings can be modified, items can be appended to the document, and it can be printed or displayed in a print preview window.

Important Note:

`new` will return an **error code** instead of a document object if there is no currently selected printer, or a printing error occurs. Always check the result of new with the `objectP()` function to make sure you have a valid Xtra instance before continuing!

Example:

The following code example creates a new document, sets the page orientation to landscape mode, creates a new page and "frame" on the page, appends an image with a caption to the document, and prints the document:

```
set doc = new(xtra "PrintOMatic")
if not objectP(doc) then exit
setLandscapeMode doc, TRUE
newPage doc
newFrame doc, Rect(0,0,getPageWidth(doc),getPageHeight(doc))
append doc, member "picture", TRUE
append doc, RETURN & "Image printed by the PrintOMatic Xtra.", TRUE
if doJobSetup (doc) = TRUE then print doc
set doc = 0
```

forget

You should never call the `forget` command directly for an instance of a Lingo Xtra. Director automatically calls `forget` when an instance of an Xtra needs to be disposed; calling `forget` yourself can lead to memory leaks or crashing.

Use the following syntax to explicitly get rid of a PrintOMatic document when you are done using it:

```
set doc = 0
```

Where `doc` is the PrintOMatic document you want to dispose of. Explicitly disposing of documents is optional, since Director will automatically get rid of the object when it's no longer referenced anyway.

reset

Syntax: `reset` *document*

The reset command is used to reset an instance of a PrintOMatic document. All the contents of the document and any existing pages are deleted. The page settings, such as margins and page orientation, as well as the progress dialog settings, are maintained; they are NOT reset to default values.

Page Setup and Job Setup Dialogs

The following commands are used to display the Page Setup and the print Job Setup dialog boxes for the user:

`doPageSetup`	Presents the Page Setup dialog
`doJobSetup`	Presents the Print Job Setup dialog

Calling these routines is optional; if they are not called, the document or print job will be set up with default values.

doPageSetup

Syntax: `doPageSetup(`*document*`)`

Returns: `TRUE` if the user clicks "OK" in the page setup dialog
`FALSE` if the user clicks "Cancel" in the page setup dialog

The `doPageSetup` function displays the Page Setup dialog for a document. This function must be called on an empty document, before any elements are added to it. `doPageSetup` returns `TRUE` if the user clicks the "OK" button in the dialog box, or `FALSE` if the user clicks "Cancel". You can call the `getLandscapeMode` routine before and after `doPageSetup` to determine if the user changed the page orientation in the Page Setup dialog box.

doJobSetup

Syntax: `doJobSetup(`*document*`)`

Returns: `TRUE` if the user clicks "Print" in the job setup dialog
`FALSE` if the user clicks "Cancel" in the job setup dialog

The `doJobSetup` function displays the job setup dialog for a PrintOMatic document. This function should be called right before printing. If `doJobSetup` returns `TRUE`, the user clicked the "Print" button, and printing should proceed. If `doJobSetup` returns `FALSE`, the user clicked "Cancel", and you should not print the document. This function cannot be called on an empty document.

Example:

This is the recommended way of calling `doJobSetup` right before printing a PrintOMatic document:

```
if doJobSetup (doc) = TRUE then print doc
```

Getting and Setting Document Attributes

The following commands are used get and set the attributes of a PrintOMatic document:

`setDocumentName`	Sets the name of the document
`setLandscapeMode`	Sets landscape or portrait orientation
`getLandscapeMode`	Returns the landscape mode of the document
`setMargins`	Sets the margins of the document
`getMargins`	Returns the margin settings of the document
`setPrintableMargins`	Sets the document margins to the maximum area the printer can print
`getPageWidth`	Returns the width between left and right margins
`getPageHeight`	Returns the height between the top and bottom margins
`getPaperWidth`	Returns the width of the physical paper
`getPaperHeight`	Returns the height of the physical paper

Routines that alter the size or orientation of a document, such as `setLandscapeMode`, `setMargins`, and `setPrintableMargins`, can only be called when your document is empty.

setDocumentName

Syntax: `setDocumentName` *document, name*

This command sets the name of a PrintOMatic document, which is displayed in the print progress dialog as the document prints. If background printing is enabled, this document name is also displayed by PrintMonitor (Mac) or Print Manager (Windows) as your document prints in the background.

setLandscapeMode

Syntax: `setLandscapeMode` *document, trueOrFalse*

This command switches the page orientation of a PrintOMatic document between landscape and portrait orientation. Since this method changes the whole coordinate system of the document, your document must be empty when you call setLandscapeMode. You can call reset on your document beforehand just to make sure.

In Windows, this method works in a very straightforward manner: call it, and the landscape mode changes.

Unfortunately, it's an entirely different story on the Macintosh. The *only* safe way to change the page orientation on the Mac is by showing the Page Setup dialog and letting the user manually select landscape mode. While showing a Page Setup dialog may be fine for normal software applications such as Word, it's often unacceptable in the context of a multimedia production.

To get around this, PrintOMatic for the Macintosh relies on a "printer database" to store default landscape and portrait page setups for the most common Macintosh printers. This printer database consists of a set of 'PHDL' resources located in the same file as the PrintOMatic Xtra. This database is used by the setLandscapeMode method in the Macintosh version of PrintOMatic.

If the currently selected printer is not found in the printer database, the user is asked to MANUALLY create default Page Setups for landscape and portrait modes, and those settings are saved and added to the database. This is done through a series of prompt dialogs presented automatically by PrintOMatic when you call setLandscapeMode for an unknown printer.

These user-configured "custom entries" to the printer database are stored in a file called "PrintOMatic Preferences" in the Preferences folder on the user's hard disk. Subsequent calls to setLandscapeMode on the same computer, with the same printer selected, won't present any annoying dialogs.

What the presence of this "printer database" means is that when you change the landscape mode on the Macintosh, all the other Page Setup settings such as scaling, font substitution, etc., will also revert to those found in the printer database. This is important if the user has changed any of these settings (during a call to `doPageSetup`) before setLandscapeMode is called.

getLandscapeMode

Syntax: `getLandscapeMode(`*document*`)`

Returns: `TRUE` if the document has a landscape orientation, otherwise `FALSE`

This command retrieves the page orientation of a PrintOMatic document. `getLandscapeMode` returns `TRUE` if the document has a landscape (horizontal) orientation.

setMargins

Syntax: `setMargins` *document*, *marginRect*

This command sets the margins of a PrintOMatic document. The *marginRect* parameter is in the form of a Lingo Rect. Values are specified in the format Rect(*left*, *top*, *right*, *bottom*). The measurements are in points (72 points to the inch). Since this method changes the whole coordinate system of the document, your document must be empty when you call setMargins. Call `reset` on your document beforehand just to make sure.

Example:

The following example creates a new document and sets the margins to two inches (144 points) on the left, and one inch (72 points) on all other sides.

```
set doc = new(xtra "PrintOMatic")
if not objectP(doc) then exit
setMargins doc, Rect(144,72,72,72)
```

getMargins

Syntax: `getMargins(`*document*`)`

Returns: the document margins, in Lingo `Rect` format

This command retrieves the margins of a PrintOMatic document. The return value is a Lingo Rect, but the `Rect` itself is not a valid rectangle; rather, the `top`, `left`, `bottom`, and `right` values of the `Rect` are used to return the margin width on each of those sides. The margin measurements are in points (72 points to the inch).

setPrintableMargins

Syntax: `setPrintableMargins` *document*

This command sets the margins of a PrintOMatic document equal to the maximum printable area supported by the current print settings. Since this command changes the whole coordinate system of the document, your document must be empty when you call setMargins. Call `reset` on your document beforehand just to make sure. After calling `setPrintableMargins`, you can use the `getMargins` routine to check the results.

getPageWidth

Syntax: `getPageWidth(`*document*`)`

Returns: the distance between the left and right margins, in points

This function returns the width of a document's "live area", the distance between the left and right margins of the document.

Example:

It is often convenient to retrieve the dimensions of the live area of a document and store them in Lingo variables for use in subsequent calculations. The following example creates two linked frames on the page, for formatting printed output into two columns.

```
-- create two new frames on the page
-- with 1/2 inch (36 points) in between
set w = getPageWidth(doc)
set h = getPageHeight(doc)
newFrame doc, Rect(0,0,(w/2)-18,h), TRUE
newFrame doc, Rect(0,0,(w/2)+18,h), TRUE
```

getPageHeight

Syntax: `getPageHeight(`*document*`)`

Returns: the distance between the top and bottom margins, in points

This function returns the height of a document's "live area", the distance between the top and bottom margins of the document.

Example:

It is often convenient to retrieve the dimensions of the live area of a document and store them in Lingo variables for use in subsequent calculations. The following example creates two linked frames on the page, for formatting printed output into two columns.

```
-- create two new frames on the page
-- with 1/2 inch (36 points) in between
set w = getPageWidth(doc)
set h = getPageHeight(doc)
newFrame doc, Rect(0,0,(w/2)-18,h), TRUE
newFrame doc, Rect(0,0,(w/2)+18,h), TRUE
```

getPaperWidth

Syntax: `getPaperWidth(`*document*`)`

Returns: the width of the physical paper, in points

This function returns the width of the physical piece of paper that the current document is configured to print on.

getPaperHeight

Syntax: `getPaperHeight`(*document*)

Returns: the height of the physical paper, in points

This function returns the height of the physical piece of paper that the current document is configured to print on.

Adding Pages and Setting the Current Page

The following routines are used to add new pages to your document, or set the "current" page, where subsequent text and graphic elements will be placed:

`newPage`	Create a new page
`setPage`	Set the "current" page

About the Master Page

Page 0 of each document is the document's **master page**. The master page is the default "current" page of a newly created document, or one that has been cleared by a call to `reset`. Any text or graphic element added to the master page will be drawn on every page of the document, beneath the contents of each individual page. Calling `setPage` with a value of 0 will set the master page as the current page of the document.

newPage

Syntax: `newPage`(*document*)

Returns: the page number of the newly created page

The `newPage` function adds a page to the PrintOMatic document and makes it the current page. It returns the page number of the newly created page.

NOTE: If you want your document to print, it must have at least one page. A common mistake is inadvertently add a number of elements to the *master page* — instead of page 1 — of a document by forgetting to call `newPage` beforehand. The resulting document will not print because it contains no body pages.

setPage

Syntax: `setPage` *document*, *pageNumber*

The `setPage` command sets the "current" page of the document, where subsequent text and graphic elements added to the document will be placed. If the *pageNumber* value is greater than the number of pages currently in the document, new pages will be added, and the requested page will still become the "current" page.

Calling `setPage` with a value of 0 will set the **master page** to be the current page. Any text or graphic element added to the master page will be drawn on every page of the document, beneath the contents of each individual page.

Setting Text and Graphic Attributes

The following routines are used to set the default text and graphic attributes of a document. These attributes determine the color, font, etc. of subsequently added document elements:

`setTextFont`	Sets the default text font for non-styled text data
`setTextSize`	Sets the default text size for non-styled text data
`setTextStyle`	Sets the default text style for non-styled text data
`setTextJust`	Sets the line justification of text elements
`setTextLineSpacing`	Sets the line spacing of text elements
`setColor`	Sets the default color of text and graphic elements
`setGray`	Sets the default gray value of text and graphic elements
`setLineWeight`	Sets the line weight of stroked graphic elements

setTextFont

Syntax: `setTextFont` *document*, *fontName*

Returns: `TRUE` if the requested font was available
`FALSE` if the requested font could not be found

This command sets the text font that will be applied to non-styled text data (such as strings) that are subsequently appended to the PrintOMatic document. If the requested font is not available, the default font (Geneva on the Macintosh, Arial on Windows) is used.

Note that the current font set using setTextFont will be overridden by the fonts, sizes and styles within styled text field cast members that you subsequently append to your document. The default font used for non-styled text data after appending a styled text field will be the last font contained within the text field.

setTextSize

Syntax: `setTextSize` *document*, *fontSize*

This command sets the text size that will be applied to non-styled text data (such as strings) that are subsequently appended to the PrintOMatic document.

setTextStyle

Syntax: `setTextStyle` *document*, *styleString*

This command sets the text style that will be applied to non-styled text data (such as strings) that are subsequently appended to the PrintOMatic document. The names of the style values correspond exactly to the Director "textStyle" property. Possible values for text styles are:

```
normal
plain
bold
italic
underline
```

On the Macintosh, the following additional styles are available:

```
outline
condense
extend
shadow
```

All values except `normal` or `plain` are added together in a call to `setTextStyle`, so you can combine a number of styles together in a single call.

Example:

The following example creates a PrintOMatic document, sets the default font to bold italicized 10 point Helvetica, and prints a short text string:

```
set doc = new(xtra "PrintOMatic")
if not objectP(doc) then exit
setTextStyle doc, "bold, italic"
setTextFont doc, "Helvetica"
setTextSize doc, 10
append doc, "Some sample text to print out."
if doJobSetup(doc) then print doc
set doc = 0
```

setTextJust

Syntax: `setTextJust` *document*, *justCode(s)*

The setTextJust command sets the line justification applied to all subsequently added text elements. Possible values are:

```
left
right
centered
```

Justification applies to an entire text block (or the set of linked text blocks), so if you set justification to "right" and call `append` for a block that was previously centered, the justification of the entire block will be changed to right-justified as the new text is added.

setTextLineSpacing

Syntax: `setTextLineSpacing` *document*, *spacing*

Sets the line spacing of text elements

setColor

Syntax: `setColor` *document*, *red*, *green*, *blue*

Sets the default color of text and graphic elements

setGray

Syntax: `setGray` *document*, *grayLevel*

Sets the default gray value of text and graphic elements

setLineWeight

Syntax: `setLineWeight` *document*, *lineWeight*

Sets the line weight of stroked graphic elements

Drawing Graphic Elements

The following routines are used to draw graphic elements on the pages of documents:

`drawRect`	draws a filled or stroked rectangle
`drawLine`	draws a line
`drawRoundRect`	draws a filled or storked round rect
`drawOval`	draws a filled or stroked oval
`drawText`	draws a line of text
`drawPicture`	draws a picture from the cast or from disk
`drawStagePicture`	draws a picture of the Stage contents

drawRect

Syntax: `drawRect` *document*, *rect*, *filled*

Draws a rectangle on the current page. If the *filled parameter is* `TRUE`*, the rectangle is filled using the current color. Otherwise, the rectangle is stroked using the current line weight and color.*

Example:

```
drawRect doc, Rect(0,0,500,100), TRUE
```

drawLine

Syntax: `drawLine` *document*, *startPoint*, *endPoint*

Draws a line on the current page from *startPoint* to *endPoint*. The line *is stroked using the current line weight and color.*

Example:

```
drawLine doc, Point(0,100), Point(getPageWidth(doc),100), TRUE
```

drawRoundRect

Syntax: `drawRoundRect` *document*, *rect*, *cornerRadius*, *filled*

Draws a rounded-corner rectangle on the current page, using the corner radius specified in *cornerRadius*. If the *filled parameter is* `TRUE`*, the rounded rectangle is filled using the current color. Otherwise, the rounded rectangle is stroked using the current line weight and color.*

Example:

```
drawRoundRect doc, Rect(0,0,500,100), 25, FALSE
```

drawOval

Syntax: `drawOval` *document*, *bounds*, *filled*

Draws an oval on the current page, bounded by the rectangle specified in *bounds*. If the *filled parameter is* `TRUE`*, the oval is filled using the current color. Otherwise, the oval is stroked using the current line weight and color.*

Example:

```
   drawOval doc, Rect(0,0,100,100), FALSE
```

drawText

Syntax: `drawText` *document*, *text*, *location*

Draws a line of text on the current page, using the current text font, size, style, and justification. The justification specified using `setTextJust` determines how the text is aligned relative to the point specified in *location.*

Example:

```
drawText doc, "A little bit of text.", Point(100,50)
```

drawPicture

Syntax: `drawPicture` *document*`, [` *fileName* `|` `member` *castMem* `], [` *rect* `|` *topLeftPoint* `]`

Draws a bitmapped or PICT cast member, EPS, PICT or BMP file from disk on the current page*. If a destination rect* is specified, the picture will be sized to fit within the rectangle without distortion. If a *topLeftPoint* is specified, the image will be drawn at 100% size from the specified point.

Examples:

```
drawPicture doc, member "image", Point(100,50)
drawPicture doc, the pathName&"image.eps", Rect(0,0,100,100)
```

drawStagePicture

Syntax: `drawStagePicture` *document*, `[` *rect* `|` *topLeftPoint* `],` `[` *clipRect*`],` *grabOffscreen*

Places a screen shot of the stage contents on the current page (or MIAW contents if `drawStagePicture` is called from a movie-in-a-window). *If a clipRect* is specified, only that portion of the stage or MIAW will be grabbed. Note that `drawStagePicture` takes a "faithful" screen grab of the stage contents, including the cursor, sprite "trails", and any other windows that might overlap the stage. However, if *grabOffscreen* is `TRUE`, the stage picture will be grabbed from Director's off-screen buffer instead, and these extraneous elements will not be included in the resulting picture.

Examples:

```
drawStagePicture doc, Point(0,0)
drawStagePicture doc, Point(0,0), Rect(0,0,100,300), TRUE
```

Creating Frames and Appending their Contents

The PrintOMatic Xtra uses a "frames" metaphor for creating areas on the page where text or graphics can be flowed. The basic procedure is to create one or more "linked" frames on a page of a document, then append items to the newly created frames. The following commands are used to create frames and append their contents.

`newFrame`	creates a new frame on the current page
`append`	appends contents to the current frame
`appendFile`	appends the contents of a text or graphic file to the current frame
`getInsertionPoint`	gets the location of the insertion point within the current frame

newFrame

Syntax: `newFrame` *document*, *rect*, *linkedToPrevious*

The `newFrame` command adds a new "frame" to the current page of the document, and makes it the "current" frame. Text and graphic items can be flowed into the current frame using the `append` and `appendFile` commands.

The *linkedToPrevious* parameter determines whether the contents of the previous frame will flow into the new frame once the previous frame is filled. When you create a new frame that is not *linkedToPrevious,* you can no longer append items to the previously "current" frame. Also note that you cannot link frames between the master page and a body page.

append

Syntax:

`append` *document*, `member` *whichCastmember* [, ... , *autoAppend*]
`append` *document*, `member` *whichCastmember* `of castLib` *whichLib* [, ... , *autoAppend*]
`append` *document*, `castLib` *whichCast* [, ... , *autoAppend*]
`append` *document*, `sprite` *whichSprite* [, ... , *autoAppend*]
`append` *document*, *string* [, ... , *autoAppend*]
`append` *document*, *list* [, ... , *autoAppend*]

The `append` command appends one or more items to the current "frame" of a PrintOMatic document. If no current frame exists in your document, the PrintOMatic Xtra will attempt to create a "default" frame for you, which is the width and height of the page, minus the current margins.

The *autoAppend* parameter, which is always the last parameter in a call to `append,` controls whether or not new pages will be created "on the fly" if the PrintOMatic Xtra runs out of space on the last "frame" in your document. If this parameter is set to `TRUE`, new pages will be created with the same "frame" layout as the last page of your document. PrintOMatic will create as many new pages as are necessary to flow all the specified elements into your document . If this parameter is not specified when you call `append`, its value defaults to `TRUE`.

AutoAppending is not allowed when flowing items into a frame on the master page of a document.

type of object	what gets appended
text field cast member	the text of the field, using the specified fonts and styles
rich text cast member	the bitmap image of the cast member, including anti-aliasing
bitmap cast member	the cast member graphic
PICT cast member	the cast member graphic
cast library	all printable cast members in the library, in cast sequence
sprite	the cast member of the sprite
text string	the text string, in the default font (Geneva 10pt on Macintosh, Arial 10pt on Windows)
list	the elements in the list

Example:

The following example creates a PrintOMatic document, sets the document name, creates the first page, adds a frame to it, and appends a number of items to the document, and prints it:

```
set doc = new(xtra "PrintOMatic")
if not objectP(doc) then exit
setDocumentName doc, "My Example Document"
```

```
newPage doc
newFrame doc, Rect(0,0,getPageWidth(doc),getPageHeight(doc)
append doc, sprite 1, TRUE
append doc, [member "image", member "caption", sprite 5], TRUE
append doc, castLib "printout", TRUE
if doJobSetup(doc) then print doc
set doc = 0
```

appendFile

Syntax: appendFile *document*, *fileName* [, *...* , *autoAppend*]

The `appendFile` command appends one or more text or graphics files to the current "frame" of a PrintOMatic document. If no current frame exists in your document, the PrintOMatic Xtra will attempt to create a "default" frame for you, which is the width and height of the page, minus the current margins.

For details on how to use the *autoAppend* parameter, please see the for the `append` command.

The following file formats are supported by `appendFile`. The actual format of the file will be auto-detected by the `appendFile` command.

file type	notes
plain text	Normal ASCII format text, with or without DOS line-feed characters
styled text	(Macintosh only) Macintosh ASCII text file with a 'styl' resource, such as files created by SimpleText
EPS	Encapsulated PostScript file, with or without a preview image
PICT	Macintosh PICT format file (only raster PICT files supported on Windows)
BMP	(Windows only) BMP files of any bit depth

Notes on Printing EPS Files

You should avoid using EPS files if you want your printing code to work reliably with all types of printers. Many, many popular printers attached to Macintosh and Windows PC's DO NOT support PostScript printing. The output that PrintOMatic generates on these types of printers can vary from low-resolution bitmaps to placeholder boxes to nothing at all.

Assuming you decide not to heed this warning, here are some tips that may improve your success.

Many applications that generate EPS files allow you to create files in "ASCII" or "binary" format. PrintOMatic prints ASCII format PostScript files MUCH more reliably than binary files. Under some conditions, PrintOMatic will print binary PostScript files just fine. However, certain types of printer connections work very poorly with binary PostScript. Specifically, serial printers that use XON/XOFF flow control often mistake binary data for flow control codes, and will seriously garble or crash your print job.

Some types of PostScript files, notably those generated by using the "print to disk" feature of the LaserWriter driver, don't contain "bounding box" information. PrintOMatic needs bounding box information to determine the size of a PostScript image for placement on the page, and will generate an error if this information can't be found.

You can manually add bounding box information to PostScript files using a text editing program. Insert the following line of text into the file somewhere between `the !%PS-Adobe-3.0` and `%%EndComments` lines at the beginning of the file, substituting the width and height of the page (in points) for the 'x' and 'y' values:

```
    %%BoundingBox: 0 0 x y
```

Finally, keep in mind that PostScript is a programming language with a broad set of features, some of which are supported differently by different printers. This makes EPS files MUCH more prone to printing errors and incompatibilities than other file formats, such as PICT or BMP. This is another good reason to avoid using EPS files if at all possible.

getInsertionPoint

Syntax: `getInsertionPoint(`*document*`)`

Returns: A string in the format "page, x, y", or `VOID` if there is no insertion point.

The `getInsertionPoint` function returns the page number and coordinates at which the next `append` or `appendFile` command will insert new content into the document. This can be useful for deciding when to manually break pages, or allow you to place graphics in the margins of a document next to accompanying text.

Example:

The following code checks for an insertion point, and if there is one, places a graphic in the margin next to the insertion point:

```
set the itemDelimiter = ","
set insPt = getInsertionPoint(doc)
if stringP(insPt) then
   drawPicture member "dingbat", Point(-10, integer(item 3 of insPt))
end if
```

Customizing the Print Progress Window

The following routines are used to customize the contents of PrintOMatic's Print Progress window. The `setProgressMsg` and `setProgressPict` routines are available to registered users of PrintOMatic only. Attempting to use these routines on an unregistered copy will result in an error.

`setProgressMsg`	puts a customized text message in the progress window
`setProgressPict`	puts a customized bitmap image in the progress window
`setProgressLoc`	sets the location of the progress window
`clearProgressLoc`	resets the location of the progress window to centered on the screen

setProgressMsg

Syntax: setProgressMsg *document*, *message*

The `setProgressMsg` command puts a customized text message into the Print Progress window. This command is available to registered users of PrintOMatic only. Attempting to use this command on an unregistered copy will result in an error.

The default message in the print progress window is:

```
   Printing document "<document name>"
```

You can change the document's name using the `setDocumentName` command.

setProgressPict

Syntax: setProgressPict *document*`,` `member` *pictCastMember*

The `setProgressPict` command puts a customized image into the Print Progress window. This command is available to registered users of PrintOMatic only. Attempting to use this command on an unregistered copy will result in an error.

setProgressLoc

Syntax: setProgressLoc *document*`,` *topLeftPoint*

The `setProgressLoc` command sets the location of the top left corner of the print progress dialog displayed during printing. If you want to hide the print progress dialog, position it off the screen using this method. Note that `setProgressLoc` uses global (screen) coordinates for positioning, not stage coordinates.

clearProgressLoc

Syntax: clearProgressLoc *document*

The `clearProgressLoc` command resets the location of the print progress dialog to its default location, centered on the screen. Good to use if you want to show the progress dialog again after setting it offscreen with `setProgressLoc.`

Printing and Print Preview

The following commands are used to print or display a print preview of a PrintOMatic document, or just about anything else you want to print out:

`printPreview`	displays an on-screen print preview of the requested item(s)
`print`	prints the requested item(s) to the currently selected printer

printPreview

Syntax: `printPreview(`*document*`)`
`printPreview(`*any*` [, ... ])`

Returns: `TRUE` if the user previews all the pages in the document
`FALSE` if the user cancels the print preview

This function displays an on-screen facsimile of the output that the `print` command will generate when it is passed the same set of parameters. `printPreview` will display a preview of a document as well as sets of strings, sprites, cast members, lists, etc. See the `print` command for a complete list of the elements that can be previewed.

Typing any key or clicking in the preview window advances to the next page; typing command-period (Escape on Windows) cancels the preview without displaying all the pages. When the document can't be displayed on the main screen at 100% size (which is most of the time, unless you have a big monitor), the page is scaled to fit.

print

Syntax:

`print member` *whichCastmember* [,...]
`print member` *whichCastmember* `of castLib` *whichLib* `[, ... ]`
`print castLib` *whichCast* [, ...]
`print sprite` *whichSprite* [, ...]
`print` *string* [, ...]
`print` *list* [, ...]
`print` *document*

The `print` command is an extension to Lingo provided by the PrintOMatic Xtra. This command is used to print one or more items to the currently selected printer. The following Director objects are printable:

type of object	what gets printed
text field cast member	the text of the field, using the specified fonts and styles
rich text cast member	the bitmap image of the cast member, including anti-aliasing
bitmap cast member	the cast member graphic
PICT cast member	the cast member graphic
cast library	all printable cast members in the library, in cast sequence
sprite	the cast member of the sprite
text string	the text string, in the default font (Geneva 10pt on Macintosh, Arial 10pt on Windows)
list	the elements in the list
document	the contents of the PrintOMatic document

Examples:

You can print a number of objects together with a single call to the `print` command:

```
   print "Printed Output" & RETURN , member "image" , member "someText"
```

One of the most powerful uses of the print command is to assemble all the elements of a document into a single cast library, and print the entire cast library with one line of Lingo:

```
   print castLib "document"
```

printStage

Syntax: `printStage [(`*grabOffscreen*`)]`

Returns: `TRUE`, if printing was successful, `FALSE` if printing was canceled.

This function prints the contents of the Stage (or Authorware presentation window) with one command. This routine uses a “best fit” method to position the stage contents on the printed page. If the stage is wider than it is tall, the paper orientation is automatically switched to landscape mode. If the stage is exceptionally large, it is proportionally scaled to fit on the page within one-inch margins. The output is always centered on the page.

The *grabOffscreen* parameter affects whether the image on the stage is grabbed from the screen, or Director’s offscreen buffer. The default is to grab from the offscreen buffer, which will not capture “trails”, overlapping movies-in-a-window, the mouse cursor, or other windows. If *grabOffscreen* is false, the Xtra will take a “screen shot” of the stage, including all “trails”, and any other elements that may overlap the stage.

This command is **not available in projectors** until you register your copy of PrintOMatic.

Examples:

```
printStage        -– prints from offscreen buffer

printStage(FALSE) -- prints a screen shot
```

Miscellaneous Routines

The following commands are odds and ends that didn't fit any other category:

`hideMessages`	hides the PrintOMatic Xtra's error alerts
`setPageNumSymbol`	sets the page numbering substitution symbol
`register`	registers your copy of the PrintOMatic Xtra
`setLowMemLimits`	sets the Xtra's low-memory limits

hideMessages

Syntax: `hideMessages` *document, trueOrFalse*

The `hideMessages` command will prevent PrintOMatic from displaying alert dialogs when something goes wrong. You should strive to make your code solid enough so PrintOMatic will never have to display an error alert (without invoking this command). But if you need to suppress the messages, you can do it here.

setPageNumSymbol

Syntax: `setPageNumSymbol` *document, symbol*

The `setPageNumSymbol` command selects a text symbol whose every occurrence in the document will be replaced with the current page number at printing time. This lets you number your pages by including a text element on the master page containing the symbol you define for page numbering. Call `setPageNumSymbol` with a value of "" (an empty string) to turn off this feature.

getVersion

Syntax: `getVersion(xtra "PrintOMatic ")`

Returns: the version number of the Xtra, in string form

This function returns the version number of the xtra.

Example:

```
set version = getVersion(xtra "PrintOMatic")
if version <> "1.5.5" then alert "Incompatible Xtra version."
```

register

Syntax: `register(xtra "PrintOMatic",` *serialNumber*`)`

Returns: `TRUE` if the serial number is valid

The `register` command will register the PrintOMatic Xtra, once you license PrintOMatic and obtain a valid serial number. This will enable the methods for fully customizing your print job, such as `setProgressMsg` and `setProgressPict`. It will also remove those annoying "unregistered copy" messages from your print progress dialog and printed output.

The registration process only registers your copy of PrintOMatic **temporarily**. Your serial number is stored in Director's registry and might not be stored for any longer than the current Director session. If you re-install Director, move your movie to another computer, or use PrintOMatic from a Projector, the serial number will likely be unavailable the next time you use PrintOMatic.

Therefore, you should add the registration code below to the `startMovie` handler of **every** movie that uses the PrintOMatic Xtra. This will ensure that PrintOMatic can always find its serial number when it needs to.

Example:

The following code registers your copy of the PrintOMatic Xtra for the duration of the current Director session:

```
   register(xtra "PrintOMatic", "<your serial number>")
```

setLowMemLimits

Syntax: `setLowMemLimits` *document*, *globalHeap*, *localHeap*

This feature is not yet implemented in the current version of the PrintOMatic Xtra.

Macintosh-Only Routines

The following commands are only available in the Macintosh version of the PrintOMatic Xtra. They are provided primarily for functional backwards-compatibility with the older, XObject version of PrintOMatic:

`printToPictFiles`	prints the document to a series of PICT images on disk
`draw1bitStagePicture`	draws the stage contents as a 1-bit dithered image
`loadPageSetup`	loads saved Page Setup settings from a file on disk
`savePageSetup`	saves the current Page Setup settings to a file on disk

If you are creating a cross-platform Director production, you should avoid using these routines, or make sure to only use them within blocks of Lingo that are executed when your production is running on a Macintosh computer:

```
if the machineType <> 256 then
    -- running on a Mac
    printToPictFiles doc, the pathName&"picts"
end if
```

printToPictFiles

Syntax: `printToPictFiles` *document*, *folderName*

NOTE: This is a Macintosh-only command. Do not use this command with the Windows version of the PrintOMatic Xtra.

Outputs your document to a series of PICT files on disk. If you do not provide a *folderName*, the user will be prompted to select a folder. The PICT files are sequentially titled "Page-#". The PICTs will over-write any files in the destination folder with the same name. If you only want to print certain pages to disk, present the user with the print job setup dialog using `doJobSetup` first. `printToPictFiles` will honor the start and end page set from the job setup dialog.

draw1bitStagePicture

Syntax: `draw1bitStagePicture` *document*, `[` *rect* `|` *topLeftPoint* `],` `[` *clipRect*`],` *grabOffscreen*

NOTE: This is a Macintosh-only command. Do not use this command with the Windows version of the PrintOMatic Xtra.

Places a screen shot of the stage contents, dithered to 1-bit color, on the current page (or MIAW contents if `draw1bitStagePicture` is called from a movie-in-a-window)*. If a clipRect* is specified, only that portion of the stage or MIAW will be grabbed. Note that `draw1bitStagePicture` takes a "faithful" screen grab of the stage contents, including the cursor, sprite "trails", and any other windows that might overlap the stage. However, if *grabOffscreen* is `TRUE`, the stage picture will be grabbed from Director's off-screen buffer instead, and these extraneous elements will not be included in the resulting picture.

Examples:

```
draw1bitStagePicture doc, Point(0,0)
draw1bitStagePicture doc, Point(0,0), Rect(0,0,100,300), TRUE
```

loadPageSetup

Syntax: `loadPageSetup(`*document*, *fileName*, *resourceID*`)`

Returns: `TRUE` if page settings don't match the current printer, otherwise `FALSE`

NOTE: This is a Macintosh-only command. Do not use this command with the Windows version of the PrintOMatic Xtra.

Retrieves Page Setup information stored in a file on disk. If one or more Page Setup values needs to be changed to match the currently selected printer, this command returns `TRUE`. In this case, you should probably warn the user, and display the Page Setup dialog using `doPageSetup`. Otherwise, this command returns `FALSE`.

savePageSetup

Syntax: `savePageSetup` *document*, *fileName*, *fileType*, *fileCreator*, *resourceID*

NOTE: This is a Macintosh-only command. Do not use this command with the Windows version of the PrintOMatic Xtra.

Saves the Page Setup information of the current document to a 'PHDL' resource in a file on disk. If the specified file doesn't exist, PrintOMatic creates the file, using the 4-*fileCreator* and *fileType* codes you specify. If a 'PHDL' resource with the requested *resourceID* already exists, it is replaced by the current settings.

PrintOMatic Technical Notes

You can always view the most recent Tech Notes about PrintOMatic on our web site at http://www.printomatic.com/

Tech Note:-1 Error When Printing on Windows

Updated: 8/22/99
Products: PrintOMatic Xtra 1.6.2 , PrintOMatic Lite Xtra 1.6.2
Platforms: Windows 95, Windows 98, Windows NT

When trying to print using PrintOMatic or PrintOMatic Lite for Windows, an error dialog pops up that says: "The printer could not be initialized. Error code=-1"

This the result of using an out-of-date version of PrintOMatic with Director 7.

The solution is to download the latest version of PrintOMatic or PrintOMatic Lite.

For those who are interested, some background about this problem. This error is the result of a conflict between Director 7, and version 4 and later of Microsoft's MFC class libraries, which renders MFC unable to initialize the printer.

Version 1.5.8 and later of the PrintOMatic and PrintOMatic Lite Xtras for Windows were compiled using the version 2 MFC libraries, which do not have this conflict.

Tech Note: Are Serial Numbers Cross-Platform?

Updated: 8/22/99
Products: PrintOMatic Xtra 1.6.2 , Table Xtra 1.0 , Yak Xtra 1.1 , PrintOMatic Lite Xtra 1.6.2
Platforms: 68K Macintosh, Power Macintosh, Windows 3.1, Windows 95, Windows 98, Windows NT

Yes. You only have to buy one serial number for Electronic Ink's cross-platform Xtras, such as PrintOMatic, PrintOMatic Lite, Table, and Yak. The same serial number works on both Macintosh and Windows versions of these products.

Tech Note: Blank Output on Windows

Updated: 3/22/01
Products: PrintOMatic Xtra 1.6.2 , PrintOMatic Lite Xtra 1.6.2
Platforms: Windows 98, Windows 95, Windows NT
Blank or garbled output from PrintOMatic and PrintOMatic Lite for Windows is almost always a result of two factors: 16-bit color depth, combined with out-of-date or poorly written print drivers, particularly drivers for Hewlett-Packard printers. The problem sometimes occurs with systems set to 32 bit color, in addition to 16 bits. The problem is more prevalent with larger images than small.

If you are having problems printing a LINKED cast member, consult the tech note "Linked Cast Members Don't Print"

Nearly every problem with printing true color images or screen shots with PrintOMatic can be attributed to out of date or flawed printer drivers. Just because other programs can print doesn't mean your drivers are OK. Part of the problem is that very few programs actually send 16-bit data to the printer. Photoshop, for example, always sends 24-bit data to the printer, regardless of the source bit depth.

The blank output problem most commonly occurs when using Hewlett Packard print drivers to print to a black & white HP printer. We are discovering that these are the worst print drivers in the industry, with the most poorly implemented color support. HP is virtually the **only** manufacturer whose drivers have color printing flaws that cause PrintOMatic pages to come out blank.

Some Background

PrintOMatic is one of a very few programs that actually send data to the printer at the source bit depth, which for many people running Windows 95 and NT, happens to be 16 bits per pixel.

A bit of history. In Windows 3.1 and earlier, the only "legal" bit depths for DIB data (Windows' internal format for bitmapped images) were 1, 4, 8,and 24. The format was very rigid, and very simple. When Microsoft wrote Windows NT and 95, they expanded the DIB format to include a number of new bit depths (16 bits among them), and a number of extra features for handling true-color images better.

When printer driver developers updated their Windows 3.1 drivers to work in Windows 95 and NT, many of them forgot about the expanded DIB format that these operating systems supported. Some who didn't forget simply did a poor job of implementing 16-bit support. (Developers of VIDEO drivers were much better about this.) The result was print drivers that didn't support 16-bit image data, or did so poorly, even though the operating systems now had full 16-bit color support.

Over the past couple of years, most driver developers have finally stepped into line with 16-bit image data support. As a result, getting the latest version of the drivers for your printer will often solve the "blank page" bug when printing true color images with PrintOMatic.

However, many HP print drivers, especially drivers for their black & white printers, still contain color printing flaws as described above.

Recommended Solutions

1. Update your print drivers to the latest version from HP's web site.

2. Manually change the screen depth to 8 bits (most reliable) or 24 bits.

3. Detect the screen depth using `the colorDepth`. If `the colorDepth = 16`, then alert the user and use an Xtra like The DisplayRes Xtra to switch bit depth on the fly. (A tutorial on the DisplayRes Xtra is out of the scope of this tech note.)

Tech Note: Can’t Print an Empty Document

Updated: 9/18/00
Products: PrintOMatic Xtra 1.6.2 , PrintOMatic Lite Xtra 1.6.2
Platforms: 68K Macintosh, Power Macintosh

The Problem: When trying to print a sprite using the syntax:

```
 print sprite X
```

PrintOMatic responds with an error: "Can't print an empty document."

The Solution: There is a Lingo bug in the Macintosh version of Director 7 and later that prevents this syntax from working. Instead, use the syntax:

```
 print the member of sprite X
```

Tech Note: Customizing (and Hiding) the Print Dialogs

Updated: 4/17/01
Products: PrintOMatic Xtra 1.6.2
Platforms: 68K Macintosh, Windows 3.1, Windows 98, Power Macintosh, Windows 95, Windows NT

There are three dialogs associated with the printing process in PrintOMatic. These dialogs are:

• The Page Setup Dialog
• The Job Setup Dialog
• The Print Progress Dialog

The first two dialogs can be displayed to the user or not, at your discretion. The Print Progress dialog can be hidden from the user, positioned wherever you want it on the screen, or customized with a graphic of your choice.

Pre-Requisites

Before you can customize the printing dialogs, you must be using **document objects** instead of the simple "print" commands to do your printing.

You create a document object by "creating an instance" of the PrintOMatic Xtra, and then manipulating the document object before you finally print its contents. A basic printing framework that uses a document object to print (with the normal dialogs) looks like this:

```
set doc = new (xtra "printomatic")
if not objectp(doc) then exit
append doc, member "blah"
if doJobSetup(doc) = TRUE then print doc
set doc = 0
```

Displaying the Page Setup Dialog

You must display the page setup dialog box before you add anything to your document. Once you start addding items to your document object, the page layout cannot be changed. Display the page setup dialog by adding a new line to the basic script:

```
set doc = new (xtra "printomatic")
if not objectp(doc) then exit
if doPageSetup(doc) = FALSE then exit
append doc, member "blah"
if doJobSetup(doc) = TRUE then print doc
set doc = 0
```

The "if" statement bracketing the `doPageSetup` command cancels the print job if the user clicks the "Cancel" button.

Hiding the Job Setup Dialog

It's easy to hide the job setup dialog box from your users. Simply don't call `doJobSetup()` before calling `print`:

```
set doc = new (xtra "printomatic")
if not objectp(doc) then exit
append doc, member "blah"
print doc
```

```
   set doc = 0
```

Hiding the Print Progress Dialog

To hide the print progess dialog, use the `setProgessLoc()` command to position it off the screen using **negative** coordinates:

```
   set doc = new (xtra "printomatic")
   if not objectp(doc) then exit
   append doc, member "blah"
   setProgressLoc doc, Point(-1000,-1000)
   if doJobSetup(doc) = TRUE then print doc
   set doc = 0
```

The `setProgessLoc` command is only available when you register your copy of the Xtra.

Customizing the Print Progress Dialog

You cam place your own custom bitmap into the Print Progress Dialog. Simply use the `setProgressPict()` command to specify a bitmap cast member you would like to use:

```
   set doc = new (xtra "printomatic")
   if not objectp(doc) then exit
   append doc, member "blah"
   setProgressPict doc, member "myprogresspict"
   if doJobSetup(doc) = TRUE then print doc
   set doc = 0
```

The `setProgessPict` command is only available when you register your copy of the Xtra.

Tech Note: Linked Cast Members Don’t Print

Updated: 3/22/01
Products: PrintOMatic Xtra 1.6.2 , PrintOMatic Lite Xtra 1.6.2
Platforms: 68K Macintosh, Windows 3.1, Windows 98, Power Macintosh, Windows 95, Windows NT

The Problem:

Cast members that are linked to a file on disk (especially a large or high-resolution file) don't print.

Solution:

You might have to force Director to preload the cast member using the `preloadCast` command, before you print the member.

Also, the default memory size for projectors on the Macintosh is quite small. You might have to increase the projector's memory usage in order to have enough memory for the projector to properly load the cast member for printing.

Tech Note: Macintosh PHDL Resources

Updated: 8/22/99
Products: PrintOMatic Xtra 1.6.2
Platforms: 68K Macintosh, Power Macintosh

What PHDLs Are Used For

Because Apple supplies a non-standard method for handling landscape printing mode, PrintOMatic Xtra uses an internal PHDL resource to provide information which in fact is printer driver specific. As shipped, the PHDL resource covers most known common Macintosh compatible laser printers.

If you attempt to set landscape mode on a printer which is "unknown", PrintOMatic Xtra puts up a special one-time only dialog asking for some information which it then stores in the PHDL resource of a special file named "PrintOMatic Preferences" that PrintOMatic Xtra will use in future sessions. The "PrintOMatic Preferences" file is stored in your System Folder's "Preferences" folder.

It's not so much related to *brand* of printer as it is to the printer *driver*. For example, Apple's LaserWriter driver is used for just about every PostScript printer that's out there. Printer-specific features are implemented by the Macintosh OS via .PPD files. This is nice, because it allows the same driver (with the same printer database entry) to be used for a lot of printers.

Which PHDL Resources Come with PrintOMatic?

The default PHDL resources are actually named with the driver that generated them, so you can just look at the names with ResEdit or Resourcer. These are the ones that are in there:

- Express Fax Sender
- ImageWriter
- LaserWriter
- LaserWriter 300
- CoStar LabelWriter
- Acrobat PDFWriter
- LinkSaver 6.1

You don't actually have to have the printer attached to your machine to generate these entries; you just have to have the driver selected in the Chooser. Then calling setLandscapeMode will prompt you to configure the default Landscape and Portrait page setups, and write the appropriate 'PHDL' resources to the "PrintOMatic Preferences" file.

How to Install Custom PHDLs

Using the following steps, you can not only add additional PHDL resources to PrintOMatic Xtra but you can change the default paper sizes used by PrintOMatic Xtra! Here's how to do this:

1. Install all of the printer drivers you want to create PHDL resources for into your Extensions folder.

2. Launch Director (with PrintOMatic Xtra installed in your Xtras folder.)

3. Create an instance of the PrintOMatic Xtra in Director: put new(xtra "PrintOMatic") into doc

4. Use the Chooser to select every print driver (one at a time) and call setLandscapeMode for each driver: setLandscapeMode doc, TRUE

5. Following the on-screen prompts, carefully configure the default landscape and portrait page setup. They'll be saved to the "PrintOMatic Preferences" file in your Preferences folder.

6. Repeat steps 4 and 5 until you have used every printer driver.

7. Quit Director.

Now, you need to copy the PHDL resources from the "PrintOMatic Preferences" file that has been created into a new copy of the PrintOMatic Xtra...
1. Make a backup copy of your PrintOMatic Xtra file. IMPORTANT!!!

2. Open your backup copy of the PrintOMatic Xtra file with ResEdit or Resourcerer.

3. If you want to change the default paper sizes for existing PHDL resources, delete the existing PHDL resources.

4. Open your "PrintOMatic Preferences" file with ResEdit or Resourcer.

5. Copy the desired PHDL resources from the "PrintOMatic Preferences" file into the PHDL section of the PrintOMatic Xtra backup file.

6. Save the changes to the PrintOMatic Xtra backup file.

You're done. Now try everything out by using your modified backup copy of the PrintOMatic Xtra and see if you have properly copied the PHDL resources into the Xtra. Once you have tested everything carefully, you can use your modified copy of the PrintOMatic Xtra for your shipping project!

Tech Note: Selecting Paper Sizes on Macintosh

Updated: 8/22/99
Products: PrintOMatic Xtra 1.6.2 , PrintOMatic Lite Xtra 1.6.2
Platforms: 68K Macintosh, Power Macintosh

The PrintOMatic and PrintOMatic Lite Xtras use the current printer driver to determine the default paper size for your documents. You can control this setting in a number of ways:

Selecting a Printer Driver

On Macintosh systems, you select your printer driver using the Chooser application (accessible from the Apple menu.) When you open the Chooser, you are presented with icons representing the different printer drivers that you have installed. After you select the desired driver, the Chooser will display all available printers on the right side of the dialog. Next, you click on the name of the printer that you want to use.

On Macintosh systems, selecting a printer and printer driver does NOT give you the option of selecting the default paper size. This setting is controlled explicitly from the standard Page Setup dialog.

Therefore, to allow PrintOMatic Xtra documents to be printed using paper sizes other than the default, you need to use the doPageSetup command. Using doPageSetup If you plan to make use of this PrintOMatic Xtra command, you should call it prior to adding any content to your document. This command will give the user the opportunity to select the paper size they want to use by displaying an OS standard Page Setup dialog.

Once the user has made their selection, you can determine the dimensions of the paper size that they have selected by using the getPaperWidth() and getPaperHeight() functions. In addition, you can adjust the document layout space to the maximum allowed by the current paper size setting by using the setPrintableMargins command.

Using SavePageSetup and LoadPageSetup

If you want to make a specific paper size the default, you can accomplish this on the Macintosh by using the savePageSetup and loadPageSetup commands. The savePageSetup command uses the following syntax: savePageSetup object, string fileName, string fileType, string fileCreator, int resourceID object: this should be the instance of the PrintOMatic Xtra you have created fileName: this should be an explicit path and filename for storing the page setup information fileType: this should be a four character sequence. You can use something of your own design, such as "PAGE" fileCreator: this should be a four character sequence. You can use something of your own design, such as "POMX" resourceID: this should be an integer value from 1 to 255. You can save multiple page setups into the same file, so you can use this id to identify the separate page setups.

Example:

```
    savePageSetup myDocument, the pathName & "pageSetups", "PAGE", "POMX", 1
```

You can treat this command as a function to verify that your attempt to save the page setup information was successful:

```
    put savePageSetup(myDocument, the pathName & "pageSetups", "PAGE", 
"POMX", 1) into saveSuccess
        if saveSuccess then
        -- continue on
    end if
```

It's important to note that this command saves the information currently stored in the Page Setup dialog.

So, if you want to save non-default settings, you need to first call the doPageSetup command to allow the user to select new settings.

Alternatively, you can do this yourself prior to delivering your project. You can save multiple page setups into the same file (or use separate page setup files) so you can store the settings for several paper sizes in advance.

Once you have created page setup files, you can use them with the loadPageSetup command. The loadPageSetup command uses the following syntax:

`loadPageSetup` *object*, *fileName*, *resourceID*

object: this should be the instance of the PrintOMatic Xtra you have created

fileName: this should be an explicit path and filename for storing the page setup information

resourceID: this should be an integer value from 1 to 255. You can save multiple page setups into the same file, so you can use this id to identify the separate page setups.

Tech Note: Selecting Paper Sizes on Windows

Updated: 8/22/99
Products: PrintOMatic Xtra 1.6.2 , PrintOMatic Lite Xtra 1.6.2
Platforms: Windows 95, Windows 98, Windows NT
The PrintOMatic Xtra uses the current printer driver to determine the default paper size for your documents. You can control this setting in a number of ways:

Selecting a Printer Driver

On Windows systems, you select your printer driver using the Printer Control Panel. When you open this control panel, you will be prompted with the option of selecting a printer driver (the method of this selection varies between versions of the Windows OS.) Once you have selected a printer driver, you can access the Properties for that printer driver. In the Properties dialog, you will find an option for selecting paper size.

Unlike the Macintosh OS, here you can specify the default paper size to be used by PrintOMatic Xtra. On Windows, PrintOMatic Xtra gets all of its page size and orientation information from the printer driver that is currently active. After changing your printer driver settings you need to restart Director or your runtime application so that PrintOMatic Xtra will recognize them.

Using SavePageSetup and LoadPageSetup

There is no equivalent to the savePageSetup and loadPageSetup commands for Windows. This is due to the way that the Windows OS functions.

PHDL Resources

There is no equivalent to the PHDL resources and PrintOMatic Preferences file for Windows.

Summary

In short, the only way to set the paper size to a non-default setting is to either manually change the paper size using the Printer Control Panel options or to display the Page Setup dialog using the doPageSetup command provided by PrintOMatic Xtra.

Tech Note: Serial Numbers and Registration

Updated: 1/29/00
Products: PrintOMatic Xtra 1.6.2 , Table Xtra 1.0 , Yak Xtra 1.1 , PrintOMatic Lite Xtra 1.6.2

Purchasing a Serial Number

Being a modern and low overhead company, we don't actually have product disks, manuals, and boxes to sell. Instead, you can download as many products as you want from our Web site, and **buy serial numbers** for them from our web site once you're convinced they fit your needs.

By buying serial numbers instead of boxes with disks in them, you get fully registered software right away, and you avoid paying any sales tax or shipping.

Registering Your Xtra

All of Electronic Ink's Xtras must be registered using Lingo code. You need to place a call to the Xtra's "register" routine in a place where it will run as soon as your movie is started. We recommend placing this code in your startMovie handler:

```
register xtra "[xtra name]", "[serial number]"
```

When you purchase a serial number for one of our Xtras, you will receive an email with more specific and detailed registration instructions.

The Registration Dialog Box

Some versions of our Xtras had a registration dialog box. This dialog **has been removed** in recent releases, because this registration system proved unreliable. The ONLY way to properly register an Electronic Ink Xtra is using Lingo.

Common Registration Problems

The most common registration problem is placing your registration Lingo in a place where it does not get executed.

The most common problem occurs when your `on startMovie` handler is not located in a MOVIE script cast member. To confirm the type of script member, click the "info" button at the top of the script window that contains your registration code. The Type of the script should be "Movie".

If this is not the case, you need to create a new movie script cast member for your registration code. In Director, type command-zero (Mac) or Ctrl-zero (Windows) to open a script window. If this window already contains a script, click on the "+" (plus) icon at the top left corner of the script window to create a blank script member. Check the type of the member by clicking on the "info" icon at the top of the window. The type should be "Movie". Then insert your registration code into the script window:

```
on startMovie
    register xtra "PrintOMatic", "[serial number]"
end
```

The serial number can be found in your registration email.

Registration problems can also happen if you inadvertently have more than one `on startMovie` handler. To check for duplicate `startMovie` handlers, simply perform a find in all casts for "`startMovie`". A handler that begins with `on startMovie` should only appear ONCE.

Another PrintOMatic-specific problem is having both PrintOMatic and PrintOMatic Lite Xtras installed at the same time. Since both Xtras support the `printStage` handler and other handlers, you might end up registering the lite version of the Xtra, while printing using an unregstered copy of the full version. The solution is to **only install the Xtra you are registering**: if you are using the Lite version of PrintOMatic, remove all copies of the full version from your Xtras folder, and vice versa.

Tech Note: Shockwave and PrintOMatic

Updated: 8/2/00
Products: PrintOMatic Xtra 1.6.2 , PrintOMatic Lite Xtra 1.6.2
Platforms: Windows 95, Power Macintosh, Windows 98

The PrintOMatic and PrintOMatic Lite Xtras have been created "safe for Shockwave", and are available for use and auto-download in your Shockwave movies.

This only is a brief note about using the PrintOMatic Xtra in the Shockwave environment. For complete background on creating Shockwave movies, please read the appropriate chapters of the *Using Director* manual.

In order to use Xtras in Shockwave, you must prepare your Shockwave movie to auto-download the PrintOMatic or PrintOMatic Lite Xtras when the end-user doesn't have the Xtra installed. Also, the end-user's Web browser must have the Shockwave plugin installed, and have access to www.printomatic.com, to auto-dowload the Xtras from the Internet if needed.

Modifying Xtrainfo.txt

The `XTRAINFO.TXT` file is located in the same folder as the Director application. Director uses this file to determine the "auto-downloading" information for any Xtras you want to use with Shockwave.

The version of `XTRAINFO.TXT` that ships with Director 7 does not contain the correct information about the PrintOMatic or PrintOMatic Lite Xtras. You must modify this file in order to use these Xtras with Shockwave:

1. Make sure the Director 7 application is not running.

2. Open the `XTRAINFO.TXT` file using any text editor.

3. Find the lines in the original file that look like this:

```
; PrintOMatic Lite from Electronic Ink
[#namePPC:"PrintOMatic Lite Xtra", #nameW32:"PMLITE.X32"
```

4. Replace those lines with the following text (there should be only ONE line of text per xtra, or TWO lines total):

```
; Xtras from Electronic Ink
[#namePPC:"PrintOMatic Lite Xtra", #nameW32:"PMLITE.X32",
#package:"http://www.printomatic.com/packages/PMLite"]
[#namePPC:"PrintOMatic Xtra", #nameW32:"PMATIC.X32",
#package:"http://www.printomatic.com/packages/PMatic"]
```

Preparing Your Movies

After you have modified the `XTRAINFO.TXT` files, you can add "auto-download" information to your Shockwave movies.

Do the following for every movie that uses the PrintOMatic or PrintOMatic Lite Xtra:

1. Make sure you have an active Internet connection. You will NOT be able to add Xtras to your Shockwave movie without an active connection.

2. Open the Director movie that uses the Xtra

3. Open the Movie Xtras dialog box by selecting Modify -> Movie -> Xtras from Director's Menu bar.

4. Find the Xtra you are using in the list of Xtras. In Windows, the file name for the PrintOMatic Xtra is PMATIC.X32, and for PrintOMatic Lite, it's PMLITE.X32. On the Macintosh, the Xtras are denoted by their full names.

5. If you don't see the Xtra in this list, click the "Add" button, and locate the Xtra file in the complete list of Xtras.

6. Select the Xtra in the list, and turn on the options "Include in Projector" and "Download if Needed". When you turn on "Download if Needed", Director will connect to the PrintOMatic web site to check for the downloadable Xtra "packages".

When you save your movie as a Shockwave .DCR file, it will be able to automatically download the Xtra when it is needed.

Installing Shockwave in a Web Browser

Internet Explorer will install the Shockwave plugin *automatically* when it encounter a Shockwave movie on a web page.

When Internet Explorer encounters a Shockwave movie, it will connect to Macromedia's web site to begin the download process, and launch the Shockwave Installation Wizard. Depending upon the speed of the Internet connection, this installation process can take from 1 to 20 minutes.

When a Netscape browser encounters a Shockwave movie, you will be directed to a download page at Macromedia. Click the "download" button in the first panel on this page to download the installer program. When downloading is complete, you must quit your web browser and run the installer. Then, the installer will automatically launch your browser again to complete installation.

After the installation process is complete, the web Browser can load and play the Shockwave movie.

Auto-Download

If a Shockwave movie has been properly configured, the Shockwave plugin will download any needed Xtras after loading the movie. All of Electronic Ink's Xtra packages are located at `http://www.printomatic.com/packages/` In many browsers, the Status Bar will display the message "Downloading Xtras". The end-user will be presented with a confirmation dialog describing the Xtra, and the user must confirm the installation. After the user confirms the installation, the Xtra will be placed in Shockwave's Xtras folder.

What if users decline to download the Xtra?

Users can choose NOT to download Xtras when running Shockwave movies. Your code should defend against this situation. Design your interface and coding approach to accomodate both printing and not printing.

Whenever you want to use printing functions, use the code below:

```
if the platform contains "mac" and the runMode contains "plugin"
then
  place non-PrintOMatic code here
else
  place PrintOMatic code here
end if
```

Two suggested uses of the above include when the PrintOMatic Xtra is registered and/or instantiated, and prior to any printing function being executed. The best approach for the latter example might be to change your interface. For example, if your printing function is triggered from a button, you might include the following in a sprite behavior

```
on beginSprite me
  if the platform contains "mac" and the runMode contains "plugin"
then
    sprite(me.spriteNum).locH = -500
  end if
end beginSprite
```

This will prevent the user from clicking on the button.

Similarly, the following might be a way to handle registering or instantiating the xtra:

```
on startMovie
  if not (the platform contains "mac" and the runMode contains
"plugin") then
    register xtra("printomatic", "xxxx-xxxx-xxxx")
  end if
end
```

Try It Out

You can view properly configured Shockwave versions of the PrintOMatic and PrintOMatic Lite demo movies on our web site:

PrintOMatic: http://www.printomatic.com/shockwave/pmatic/
PrintOMatic Lite: http://www.printomatic.com/shockwave/pmlite

Tech Note: Text Fields Print with Incorrect Fonts

Updated: 6/13/00
Products: PrintOMatic Xtra 1.6.2 , PrintOMatic Lite Xtra 1.6.2
Platforms: Windows 3.1, Windows 95, Windows 98, Windows NT

Symptoms:

On the Windows platform, text field cast members are not printing in the fonts you specified in the cast members.

Possible Causes:

1. You are using "embedded fonts" which are not installed in the user's system. Embedded fonts are not accessible to the OS and cannot be used for printing unless the font is also installed to the user's operating system.

2. You are using Director 7 or 8, and your cast member is not in the **first internal cast**. A bug in these versions of Director garbles font information for any text field that is not in the first internal cast. Moving your text field members to the first internal cast will fix the problem.

Tech Note: Text Members and PrintOMatic

Updated: 1/29/00
Products: PrintOMatic Xtra 1.6.2 , PrintOMatic Lite Xtra 1.6.2
Platforms: 68K Macintosh, Power Macintosh, Windows 3.1, Windows 95, Windows 98, Windows NT
Macromedia Director 5 and later have two types of cast members: text "field" cast members and rich text cast members. PrintOMatic is capable of printing both types of text members, however, this printing is handled in different ways.

While Director 5 and 6 allow you to edit Rich Text cast members as word-processing text, the ultimate format that this information is saved in is a BITMAP. While this has changed in Director 7, PrintOMatic still only assumes that it can retrieve bitmap information from RTF cast members.

In Director 5 and 6, when you protect a Director movie and bundle it into a projector, all the text information is removed from Rich Text cast members, and only the bitmap image of the text remains.

It is this bitmap image that PrintOMatic and PrintOMatic Lite are able to print. The Xtras handle an RTF member as a large bitmap, and therefore, these cast members are subject to bitmap image scaling rules:

1. If the member is wider or taller than the printable area of the page, it will be scaled to fit the page.

2. You can't "flow" the contents of an RTF member across more than one page. In order to print an RTF member on more than one page, you must break the text up into multiple RTF members, and print them by using the "append" command for each member.

Solutions

The best quality printing with PrintOMatic always comes from using text FIELD members. Consider placing your printable text into these members for best results.

Your other option is to split your RTF into multiple one-page members and print them sequentially, e.g.:

```
    print member 1, member 2, member 3
```

Tech Note: Text/RTF Members Print Too Small
Updated: 1/29/00
Products: PrintOMatic Xtra 1.6.2 , PrintOMatic Lite Xtra 1.6.2

PrintOMatic prints RTF ("text") members as bitmapped images, not as styled text. This has to do with the pre-D7 legacy of RTF members, when Director would strip out all the text from these members in Projectors.

The way that PrintOMatic handles bitmaps is to shrink them to fit on the page. As a result, long RTF members will print out at impossibly small sizes on the page.

Solutions

The best quality printing with PrintOMatic always comes from using text FIELD members. Consider placing your printable text into these members for best results.

Your other option is to split your RTF into multiple one-page members and print them sequentially, e.g.:

```
    print member 1, member 2, member 3
```

Tech Note: Using PrintOMatic in Windows 3.1

Updated: 8/23/99
Products: PrintOMatic Xtra 1.6.2
Platforms: Windows 3.1

Windows 3.1, also known as "16 bit Windows" or "Win16", is finally becoming obsolete, though it's still around. (Especially in educational and non-profit settings.)

With a 7th-generation multimedia authoring tool, it's a bit much to still expect it to support Windows 3.1, and in fact, with the release of Director 7, Macromedia quit supporting 16 bit Windows altogether, because it's just too difficult to move ahead and still maintain backwards compatibility with such an old system. As Director, Authorware and Shockwave have evolved, 16 bit versions have become harder and harder to build and maintain, and excruciating to debug. Director 7 and Shockwave 7 no longer support 16-bit Windows at all, probably for the same reasons.

In recent releases of PrintOMatic, Electronic Ink has also dropped support for Windows 3.1. The 16-bit Windows version of PrintOMatic is not available in releases after 1.5.6.

However, the 16 bit version of PrintOMatic 1.6 can still be downloaded and used. This version is **unserialized**: it does not require any serial number to work. It will safely ignore any serial number you try to register it with.

The 16 bit version of PrintOMatic has all the same commands as the current release, with the exception of the "printStage" command.

Please visit the PrintOMatic web site at http://www.printomatic.com/products.cfm to download the 16-bit version of PrintOMatic 1.5.6

Tech Note: Using PrintOMatic with Authorware

Updated: 8/23/99
Products: PrintOMatic Xtra 1.6.2
Platforms: 68K Macintosh, Power Macintosh, Windows 95, Windows 98, Windows NT

The current release of the PrintOMatic Xtra works great within Authorware 4 and 5, the versions of Authorware that have Xtras support.

There are a few limitations in working with Authorware, primarily with the data types you can print. While Director gives Xtras a lot of access to the "media" inside its cast members, Authorware has much more limited support. For example, you can't print graphics that are in Authorware, except by printing their portion of the Presentation Window, and you can only print unformatted text strings.

However, this is where PrintOMatic's support for printing graphics from disk, and formatting text fonts, sizes, and styles on the fly becomes useful.

We do not recommend using PrintOMatic Lite with Authorware, unless the only command you are interested in using is "printStage", or you only need to print simple text strings in PrintOMatic Lite's default font (Arial/Geneva).

Please download our PrintOMatic for Authorware demonstration file from our web site at ftp://ftp.printomatic.com/PMaticAWDemo.zip for complete examples of the proper syntax to use with the Xtra in Authorware.

Tech Note: Xtra Not Found Errors

Updated: 1/18/01
Products: PrintOMatic Xtra 1.6.2 , Yak Xtra 1.1 , Table Xtra 1.0 , PrintOMatic Lite Xtra 1.6.2
Platforms: 68K Macintosh, Windows 3.1, Windows 98, Power Macintosh, Windows 95, Windows NT

Symptom: When trying to register an Xtra, Director replies with an "xtra not found" error. In a projector, this may appear as a generic "Script Error".

Is the Xtra Installed?

Have you properly placed the Xtra file in Director's Xtras folder? At authoring time, all the Xtras you use need to be located in the Xtras folder, which is in the same folder as the application.

Have you created an Xtras folder for your projector?

Each projector you create needs ITS OWN XTRAS FOLDER. Projectors do not "share" Xtras with the Director application. Whenever you create a projector, you need to create an Xtras folder in the same folder as the projector, and place copies of all your third-party Xtras in this folder.

Or did you include the Xtra in your projector file?

An alternative to creating an Xtras folder for your projector is to select the Xtra file(s) you are using in the file list when you build your projector. Simply locate the Xtra files you are using, and add them right along with your .DIR and .CST files when you build your projector.

Are you using the correct Xtra file for your platform?

Macintosh and Windows Xtra files CANNOT be inter-changed across platforms. Each platform requires a completely different Xtra file.

Windows Xtra files end with a ".X32" extension. These files will NOT work on a Macintosh.

Macintosh Xtra files generally don't have extensions. They have an "X" shaped icon, and if you do a "Get Info" in the Finder on these files, they will be described as Director documents. These Xtra files CANNOT be used on Windows platforms.

Is the right PrintOMatic Xtra installed?

The PrintOMatic Xtra comes in Lite and Full versions. These versions are NOT interchangeable. If you have a serial number for the Lite version, make sure you have the Lite version installed. The file name is "PMLITE.X32" on Windows, and "PrintOMatic Lite Xtra" on the Macintosh.

If you have a serial number for the full version, make suree you have the full version installed. The file name is "PMATIC.X32" on Windows, and "PrintOMatic Xtra" on the Macintosh.

Are you using the Xtra with Shockwave on a Macintosh?

Currently, a bug prevents the use of PrintOMatic and PrintOMatic Lite with Shockwave on a Macintosh. The process will appear to work properly and will even auto-download the Xtra package if your file is set up correctly. However, whether the Xtra is installed or auto-downloaded, it will not work on a Macintosh. It will, instead, result in an "Xtra Not Found" error. See Technote: Shockwave and PrintOMatic for more information.

PrintOMatic Release Notes

1.5 Fixed a bug in the Macintosh version that would print the wrong pages if a specific page rage was selected in the Job Setup dialog. Added protective code to defend against a Director bug that returns a text size of 0 from text fields. Recompiled to remedy a bug that botched printing in Windows 95.

1.5.1 Fixed a bug that caused `append` to lock up in cases when a series of different-styled carriage returns spanned a page break. Fixed Macintosh bugs that caused print preview to mess up the Stage palette, and `drawText` to occasionally print text in funny colors.

1.5.2 Fixed a bug that caused the Windows version to crash when calling `setPage` 0.

1.5.3 Fixed a Windows bug that caused problems printing files from disk in Windows 95.

1.5.4 Internal architecture was substantially re-written to bring the code up to date for Director 6 and Authorware 4; PrintOMatic now provides limited functionality in Authorware 4 as well as Director. Changed the `reset` command so that the Page Setup and Print Progress settings are not reset along with the document contents. Added `getMargins`and `getLandscapeMode`routines. Fixed a Windows bug that caused true-color bitmaps to “wrap” by 16 pixels. Fixed Windows `drawStagePicture` bugs in 16- and 24-bit color. Cleaned up Windows print dialog code that fixes some printing anomalies (will likely stop printing leaks) in Windows 95. Changed the installation directory for the product: now ALL PrintOMatic files are placed into a sub-folder of Director's "Xtras" folder: "PrintOMatic Xtra" on the Mac, and "PMATIC" on Windows. Updated the help launcher code to be compatible with both Director 5 and Director 6 Help applications.

1.5.5 Fixed a Windows bug that caused `setLandscapeMode`routines in 1.5.4 to hang or crash. Fixed a Windows bug that prevented the global `print` command from working in 1.5.4. Rolled back to MSVC 2.0 for 32-bit Windows version due to anomalies in later versions that return a -1 error code when trying to print. Fixed a bug that prevented the splash screen from going away after the Xtra was registered. Fixed text alignment problems in `drawText` in the Macintosh version. Fixed Windows bugs that prevented adding image files from disk to a document min 1.5.4. Made PICT opcode error reporting more robust on Windows. Fixed crashing bugs in 16-bit Windows `printPreview` with EPS placeholders on the page. Fixed 16-bit Windows indexed PICT rendering glitches. Updated code to prevent drawing of PICT cast members on Windows (because Director crashes). Made `drawStagePicture` improvements that should reduce visual glitches in printing from the offscreen buffer.

1.5.6 Fixed a Windows bug that caused periodic freezing when using `append` to add pictures to frames. Fixed crashing problem with `setProgressLoc` on Windows. Added `clearProgressLoc` command to reset the location of the progress window to the center of the screen. Fixed a Windows opcode bug when converting some 1-bit PICT images.

1.5.7 Added brand-new `printStage` routine. Added brand-new registration code that allows users to enter a serial number from the Xtras menu to register the Xtra. Fixed a bug that caused some calls to the global `print` command to crash.

1.5.8 Recompiled to fix recurring -1 errors in 32-bit Windows and other anomalies.

1.5.9 Made the Xtra "Safe for Shockwave" to enable use in Shockwave 7. Fixed a corrupt message table in the Macintosh version that caused registration problems, `setPageNumSymbol` bugs, and other odd behavior.

1.6 Added support for printing Director 7 Text members directly. Modified stage-grabbing code for better 16-bit display compatibility.

1.6.1 Fixed a Windows bug that caused the captured stage image to "wrap" in 16-bit color depth. Fixed a Windows bug that sometimes caused text characters to be dropped at the beginning of pages. Changed instantiation code to allow for better error trapping when no printer is installed.

1.6.2 Fixed a Windows bug that could cause crashing when printing the Stage in 16-bit color. Updated the Macintosh version for MacOS 9 compatibility.

1.6.3 Added Windows support for rendering EPS files using the EPSPlug DLL Suite.

1.6.4 Made changes to 16- and 32-bit color bitmap printing that might improve print performance (and reduce the likelihood of blank output) with some print drivers, primarily those for HP black & white printers.

1.6.5 Mac version updated to (finally) work in Shockwave. Minor Shockwave compatibility changes to the Windows version.